FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Improving the Awareness of Technology Updates by Web Mining of Heterogeneous Online Sources

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Maximilian Schrack, BSc
Matrikelnummer 9671372

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao. Univ.Prof. Stefan Biffl
Mitwirkung: Dr. Peter Frühwirt

Wien, 26. September 2018

Maximilian Schrack

Stefan Biffl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Improving the Awareness of Technology Updates by Web Mining of Heterogeneous Online Sources

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Maximilian Schrack, BSc
Registration Number 9671372

to the Faculty of Informatics

at the TU Wien

Advisor: Ao. Univ.Prof. Stefan Biffl
Assistance: Dr. Peter Frühwirt

Vienna, 26th September, 2018

Maximilian Schrack	Stefan Biffl

Erklärung zur Verfassung der Arbeit

Maximilian Schrack, BSc
Zollergasse 32/7, 1070 Vienna

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 26. September 2018

Maximilian Schrack

Kurzfassung

Der Prozess des Software Testings ist in den letzten Jahren komplexer geworden. Die Applikationen müssen in unterschiedlichen Soft- und Hardware Umgebungen kompatibel gehalten werden, die sich zudem laufend ändern. Dies kann zum einen die Kombination aus Hardware, Software und Datenbank sein. Zum anderen muss Software heutzutage oft mit Applikationen interagieren, um beispielsweise Informationen auszutauschen oder weitere Verarbeitungsschritte in einem Prozess anzustoßen. Vor allem Schnittstellen zu anderen Systemen stellen beim Überprüfen der Kompatibilität eine große Herausforderung dar, da auf diese meist kein unmittelbarer Einfluss genommen werden kann und auch Informationen über einen neuen Release von Herstellern oft nicht aktiv kommuniziert werden. Dies kann vor allem dann eine Herausforderung darstellen, wenn eine Anpassung an einer Schnittstelle vorgenommen wurde und somit die Kompatibilität mit anderen Systemen nicht mehr gewährleistet ist. Aus der beschriebenen Problemstellung wurde die Fragestellung dieser Arbeit abgeleitet: *Inwieweit kann die Überwachung von Technologien im Hinblick auf neue Releases automatisiert werden?*

Um diese Fragestellung zu beantworten, haben wir ein System entwickelt, welches den Domainexperten bei der Überwachung von Technologien auf Updates unterstützt. Dieses System gewinnt aus verschiedenen Datenquellen Informationen, die jeweils unterschiedlich analysiert werden. In Texten, die aus Emails, RSS feeds und von Twitter extrahiert wurden, konnten mittels Natural Language Processing (NLP) Nachrichten über Technologie Updates gefunden werden. Durch die Extraktion von konkreten Release Informationen einer Online Enzyklopädie konnte ebenfalls den Prozess der Überwachung unterstützt werden. Die Analyse der Daten der Suchmaschine hat ergeben, dass ein Technologie Release einen Anstieg der Suchanfragen bewirken kann.

Im Rahmen der Arbeit wird aufgezeigt, dass sich die erschlossenen Datenquellen zur Auffindung von Technologie Releases eignen. Da der von uns entwickelte Prozess darauf aufbaut, dass ein Domainexperte die Einrichtung der Datenquellen und insbesondere die Wahl der Suchwörter, nach denen gesucht wird und welche die Ergebnisse beeinflussen können, vornimmt, konnte ein semi-automatisiertes System zur Überwachung von Technologien entwickelt werden. Der größte Vorteil des Systems besteht darin, dass der aufwändigste Teil der Technologie Überwachung - die Datenextraktion und Analyse - automatisiert werden konnte.

Abstract

The process of software testing has become more complex in recent years, especially since applications often have to work in cooperation with other technologies and therefore compatibility plays an essential role. On the one hand, a cooperation can be a connection between hardware, an operating system, and a database. On the other hand, software nowadays often has to interact with other systems, for example, to exchange information or initiate further processing steps. Interfaces to other systems, in particular, represent a major challenge in verifying compatibility, as they usually cannot be influenced directly and information about a new release from manufacturers is often not actively communicated. Above all, this can be problematic if an interface has been changed and the compatibility can no longer be guaranteed. From the described problem the main question of this thesis was derived: *To what extent can technology monitoring for new releases be automated?*

In order to answer this question, we have designed and implemented a system to support the domain expert in the process of monitoring technologies for new updates. This system extracts information from various data sources, each of which was analyzed differently. Texts extracted from email newsletters, RSS feeds and Twitter were analyzed with NLP and it turned out that it is capable of detecting release information in these texts. Online encyclopedias, from which information about previously published updates as well as preview versions of technologies could be obtained, can also help in detecting technology updates. Furthermore, the analysis of the search engine data has shown, that a technology release may cause an increase of the number of search requests extracted from Google Trends.

In summary, it can be said that the mined data sources are suitable for the process of detecting technology releases on release date as well as in advance. However, since the process we have developed relies on a domain expert to undertake the setup of the data sources, and in particular the choice of keywords that are searched and can influence the results, a semi-automated system for monitoring technologies could be developed. To be more precise, the biggest advantage of the developed system is that the otherwise very complex process of data extraction and analysis could be automated.

Contents

CHAPTER 1

Introduction

1.1 Problem Description and Motivation

Today, a software product cannot be seen as a standalone application but is often part of a big and complex hardware/software landscape and process environment. Upgrading and releasing a technology in such an ecosystem without verifying the compatibility of it's interfaces to other technology products can be associated with high risk and, in the worst case, affect the operational system's functionality. [61]

Therefore, comprehensive and time-consuming compatibility checks, conducted before the go-live of new software components, represent necessary quality assurance tasks. These checks ensure that the newly released technology is still able to interact with its environment and to minimize the possibility of undetected incompatibilities. [61, 60]

Independent Software Vendors (ISVs), which are more focused on developing software rather than hardware [21], are a driving force in the field of software development. However since the release cycle of software is steadily decreasing, keeping the company's products compatible to the system environment (e.g. underlying hardware, operating-, database- and enterprise resource planning systems, ...) forms a difficult task. This is especially the case for interfaces, which are exchanging data with third-party products. Normaly, the ISV's customer base often uses a large number of heterogeneous hardware platforms, as well as different versions of the ISV's product and the software it is interfacing with. This leads to a huge number of potential test scenarios and production environments the product has to work in. [61]

To save time for the elongated process of testing and modifying the interface of the ISV's product to the updated third party application, the ISV's goal is to detect new technology releases as early as possible.

Acquiring and providing this information to the ISV's testing and development team can be an time consuming and resource intensive task, for common software products (e.g.

Apache Tomcat, SAP, ...) and in particular for exotic ones (e.g CentOS, BS2000, ...), which are often developed for specific requirements and thus are not replaceable.

Although numerous online resources aiming at providing near-time reports on upcoming technology releases, interfaces of third party applications the ISV's products are required to interact with can change without notice [60]. To the best of our knowledge, there is no automated or highly efficient way to conduct pro-active compatibility checks, which monitor systems with regards to potentially upcoming releases. The presence of such systems, however, is desirable since the compatibility check and eventually necessary work can start earlier and therefore decreases the ISV's time pressure of testing and updating its product portfolio, especially in regard to maintenance contracts.

1.2 Research Goals and Questions

The overall goal of this thesis is to investigate the degree of automation of identifying or predicting upcoming releases of previously defined technologies (e.g. operating-, database- and enterprise resource planning systems). An automated release prediction would be a great improvement for ISVs, since an earlier identification of a technology release would relieve the ISV's time pressure of testing its product portfolio for compatibility and eventually necessary work.

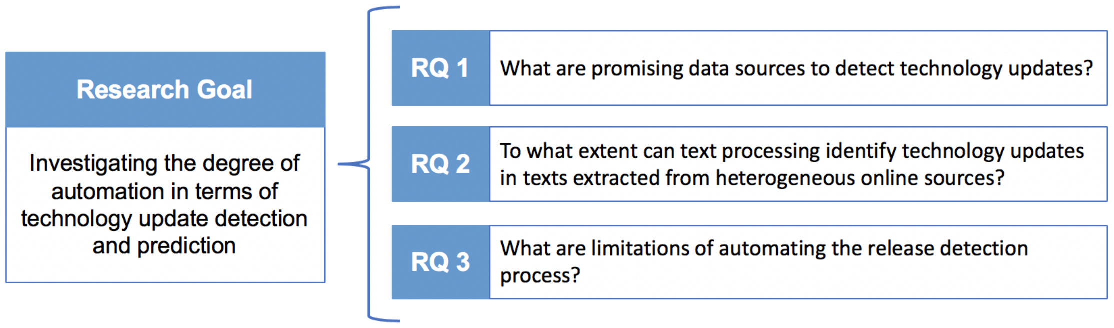

Research Goal
Investigating the degree of automation in terms of technology update detection and prediction
RQ 1
What are promising data sources to detect technology updates?
RQ 2
To what extent can text processing identify technology updates in texts extracted from heterogeneous online sources?
RQ 3
What are limitations of automating the release detection process?

Figure 1.1: Overview Research Questions

This thesis contributes three generic models for processing incoming information of heterogeneous data sources: a categorization model for online data sources, which consolidates information of the previously identified heterogeneous data sources, a decision model, which processes the collected information from the categorization model to generate an indicator for expressing the likelihood of upcoming releases and a process model for technology update detection, which should manage the data flow and input of compatibility experts.

The output of the models will be discussed by collecting and analyzing data via a prototype. The discussion should illustrate, whether fully- or semi-automated prediction of upcoming technology releases is possible. To achieve the overall goal of this thesis, we address the following research questions:

RQ 1: What are promising data sources to detect technology updates?

We first have to identify different types of relevant and trustworthy online sources (e.g. Rich Site Summary (RSS) feeds, email newsletters, websites, wikis, and so forth). Afterwards, a feasibility analysis should demonstrate whether an automated extraction method for each of these online sources can be developed. In order to consolidate and save the extracted information from heterogeneous online sources in an appropriate way, an categorization model for online data sources will be developed.

RQ 2: To what extent can text processing identify technology updates in texts extracted from heterogeneous online sources?

Based on the extracted information from heterogeneous online sources found by answering RQ 1, various approaches for text processing have to be examined to achieve the most human-like interpretation of the extracted data. This provides the foundation for developing a decision model for technology update notification, which computes the likelihood of an upcoming technology update by analyzing and combining the extracted information. In addition, the developed model should be capable of handling input by a compatibility expert. This input should communicate the system, whether the calculated update warning was reasonable or not. Furthermore, if the alert was false, the system should be capable of adapting it's parameters which indicated the update warning.

RQ 3: What are limitations of automating the release detection process?

RQ 3 sheds light on which steps of release detection of third party technologies can be performed automatically. Consequently, the analysis of the limitations should point out, which tasks still have to be performed manually by domain experts. First, we have to analyze the general process of monitoring technologies for updates to develop an own process model for technology update detection. In order to answer RQ 3, the capabilities and the actual degree of automation of the developed models will be discussed. Therefore, the categorization model for online data sources developed in RQ 1 and the decision model for technology update notification developed to answer RQ 2 will be integrated in the process model for technology update detection. The output is a prototype, which will then collect and analyze data.

By answering these research questions, on the one hand this thesis contributes to the research community of NLP by analyzing the potential of NLP to detect release messages in extracted texts. On the other hand, by analyzing the data extracted from Google Trends with various statistical methods, the results of this thesis contribute to the research community which analyzes the potential of search engine data for trend detection.

1.3 Methodological Approach

To address the research goals, the thesis' methodological approach is divided into the following six steps:

1. **Literature Review**

   An initial literature review concerning the identification of trustable information, the application of Web mining to gather this information and the consolidation of the extracted data from heterogeneous data sources must be performed. Furthermore, the various aspects of NLP for analyzing the keywords' context of found texts must be collected to complement the theoretical basis for the implementation work of the thesis.

2. **Identification of Data Sources**

   Afterwards, an extensive research of relevant and secure online information sources will take place. Presumably, the result will be a collection of email newsletters, Wikis, relevant forums, RSS feeds, and so forth. In addition, platforms (s.a. Google Trends and Bing Trends) which offer analysis of search behaviour will be considered as well. These online sources form a basis of data sets, which are proceeded in the following steps.

3. **Development of a generic Categorization-, Decision- and Process Model**

   For processing the incoming information of heterogeneous data sources, three models must be developed:

   - A **Categorization Model** for storing data from heterogeneous data sources. This data model is expected to be based on a relational database.
   - A **Decision Model**, which is capable of consolidating the stored data and in turn outputting a software trend indicator. This indicator should point out, whether an upcoming software release for a specified product was detected.
   - A **Process Model**, which illustrates how the developed software trend framework deals with detected software trends and which options of intervening for domain experts are given.

4. **Extraction of Data**

   Based on the previously identified data sources, different data extraction methods will be evaluated and checked for suitability. Presumably, the set of data sources will be collected by crawling websites, analyzing RSS feeds and email newsletters, browsing wikis and considering search trends.

5. **Prototypical Development of a Technology Trend Detection Platform**

   In order to answer the research questions 1, 2 and 3 properly, a prototypical system will be developed which implements the concepts of the three generic models. The prototype will then collect data from various online sources and analyze them by different approaches. The prototype is developed in the form of a web application and implements the models introduced before.

6. **Data Source Discussion**

   Lastly, the applicability and capability of the prototype and its underlying categorization-, decision- and process model will be evaluated. Therefore, we first will discuss the various data sources of the categorization model to show their strengths and weakness in terms of detecting (upcoming) technology releases. Second, the various analysis methods of the decision model will be compared in order to show their efficiency. And third, the analysis of the developed process model should elaborate, whether a fully (or semi-) automated technology trend analysis is feasible.

Figure 1.2 shows an overview about the different work packages resulting from the methodological approach. The basis of the implementation work is the list of requirements derived from the research questions. The requirements form the input of the literature review and the analysis process for the data sources, the decision model and the process model.

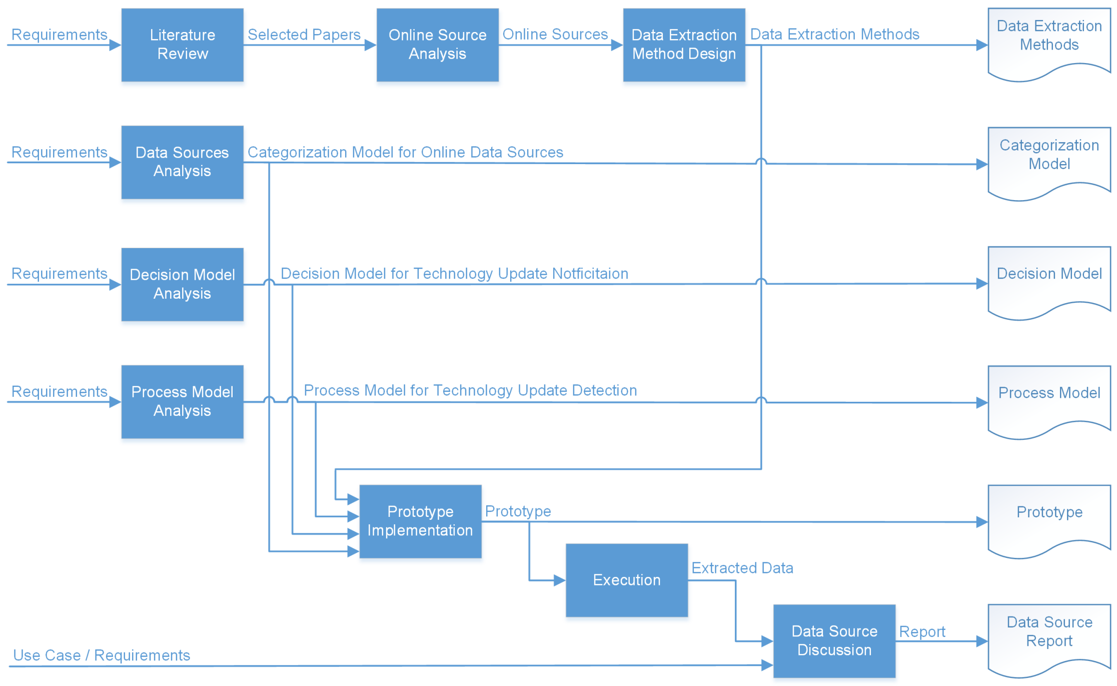


Figure 1.2: Overview Work Packages

From the literature review, we will derive knowledge about various data sources and analysis methods. In the online source analysis, we will then select various data sources, for which we will design and implement suitable data extraction methods.

The data source-, decision- and process model analyses will output three generic models, which will build - together with the data extraction methods - the basis for the implementation work of the prototype. The prototype will then collect and analyze data during its execution phase. In a last step, the results of the data analysis processes will be evaluated to answer the three research questions.

1.4 Structure of the Thesis

The purpose of Chapter 2 is to investigate various disciplines in the field of information retrieval. For this reason, the concept of Web mining is first illuminated in order to gain an overview of the different aspects of information retrieval from online sources (see Section 2.1). Next, we provide an insight into how Web pages can be tested for their credibility and which quality factors play an essential role (see Section 2.2). Furthermore, the analysis of search engines will be investigated to examine their potential for detecting technology trends (see Section 2.3).

Chapter 3 complements the previous chapter. Here, the capabilities of NLP in terms of analyzing information extracted from heterogeneous online sources to predict upcoming technology releases will be examined. Therefore, we investigate the characteristics of NLP as well as the different levels and their associated tools NLP systems have to work with.

Chapter 4 documents the developed models which build the basis of the implementation work of this thesis. Therefore, we first introduce the Process Model for Technology Update Detection in Section 4.1, which pictures the overall process of finding technology release information. Second, the Categorization Model for Online Data Sources, which contains the different types of data sources harvested to gain information from the Web, is introduced in Section 4.2. In the last Section of this Chapter, 4.3, we illustrate the process of analyzing the gained data from the Web in more detail.

In Chapter 5, we show various aspects of the implementation work done to transform the theoretical models from Chapter 4 into Java code. Therefore, we first describe both the system architecture (see Section 5.1) and the database model (see Section 5.2). Second, the different data extraction methods are explained by listings in Section 5.3. Section 5.4 shows the NLP tasks developed to extract release messages from the extracted natural language.

In Chapter 6, we discuss the data sources' potential of detecting technology releases in order to answer the research questions properly. Therefore, the discussion is categorized by data sources defined in the Categorization Model. First, we evaluate the data gained from the analysis of natural language of email newsletters, RSS feeds and Twitter posts by applying the developed NLP pipeline in Section 6.1. Second, the data source category Online Encyclopedia is analyzed in Section 6.2. Third, the capabilities of detecting an (upcoming) technology release by analyzing the number of search results extracted from Google Search are investigated in Section 6.3. In a last step, we evaluate the correlation between a change in the search volume, gained from Google Trends, and actual technology releases in Section 6.4.

In Chapter 7, we summarize our findings by first, answering the research questions in Section 7.1. In this Sections, we further discuss the differences and similarities to related work and point out the threats to validity. In Section 7.2, we introduce the next investigation steps by discussing the future research.

In the appendix (see Chapter A), we show the various findings structured by data sources. In Section A.1, newsletters we have gained data from are listed together with the amount of emails we received during the evaluation phase. In Section A.2, the list of RSS feeds we have extracted texts from are illustrated. In Section A.3, the release information extracted from Wikipedia is listed. Section A.4 gives insights into the amount of Tweets the prototype has extracted from Twitter. In the last Section of the appendix, A.5, the found peaks from analyzing the number of search requests extracted from Google Trends are listed.

CHAPTER 2

State of the Art

This chapter provides an overview of various disciplines of computer science necessary to achieve an (semi-) automated prediction of technology updates.

Therefore, we first give insights on the different aspects of Web Mining, which describe the process of gathering information from online sources in more detail (see Section 2.1). Next, we provide a brief overview on how search engines and Web users assess the quality and credibility of online sources (see Section 2.2). This is especially important to dig out the right online sources for Web mining. In addition to mining online sources (e.g. blogs, RSS feeds, email newsletters, etc.), the analysis of trends in search engines is intended to provide information on whether a technology release is to be expected (see Section 2.3). After extracting the information, the next step is to analyze the gathered content. This can be done by NLP, which is investigated in a separate chapter (see Chapter 3) because of its scope and complexity.

2.1 Web Mining

This section gives an overview about how the tools of Web mining can be used to extract information from online sources to gain knowledge about a specific topic. Therefore, we first compare various definitions of Web mining (see Section 2.1.1) and categorize the different use cases of Web mining (see Section 2.1.2). We further investigate the according subtasks (see Section 2.1.3) and the challenges Web mining has to face (see Section 2.1.4). To conclude this section, we provide insights on tools currently available on the market and compare the advantages and disadvantages (see Section 2.1.3).

2.1.1 Definition

Web mining is defined as the process of discovering and extracting useful information from online sources and is also known as Web harvesting, Web framing and Web scraping

[31, 39]. Since the idea of Web mining is based on data mining, Srivastava et. al. describe Web mining as *"(. . .) the application of data mining techniques to extract knowledge from Web data, including Web documents, hyperlinks between documents, usage logs of websites, etc."* [52, p. 399]. Etzioni describes Web mining more technical as process of automatically extracting information from Web documents/services and, at the same time, unveiling general patterns at specific Web pages as well as across multiple pages to discover potentially useful data or knowledge from online sources [18]. Berent et. al. focus in their paper more on the categorization of Web mining and define it as follows:

"Web mining is the application of data mining techniques to the content, structure, and usage of Web resources. This can help to discover global as well as local structure within and between Web pages. (..) Web mining is an invaluable help in the transformation from human understandable content to machine understandable semantics." [4, p. 266]

2.1.2 Categorization

According to the definition of Web mining from Etzioni [18], Srivastava et. al. [52] and other works, such as [39], [41] and [33], Web mining can be categorized in the following three not clear-cut types: Web content mining, Web structure mining and Web usage mining.

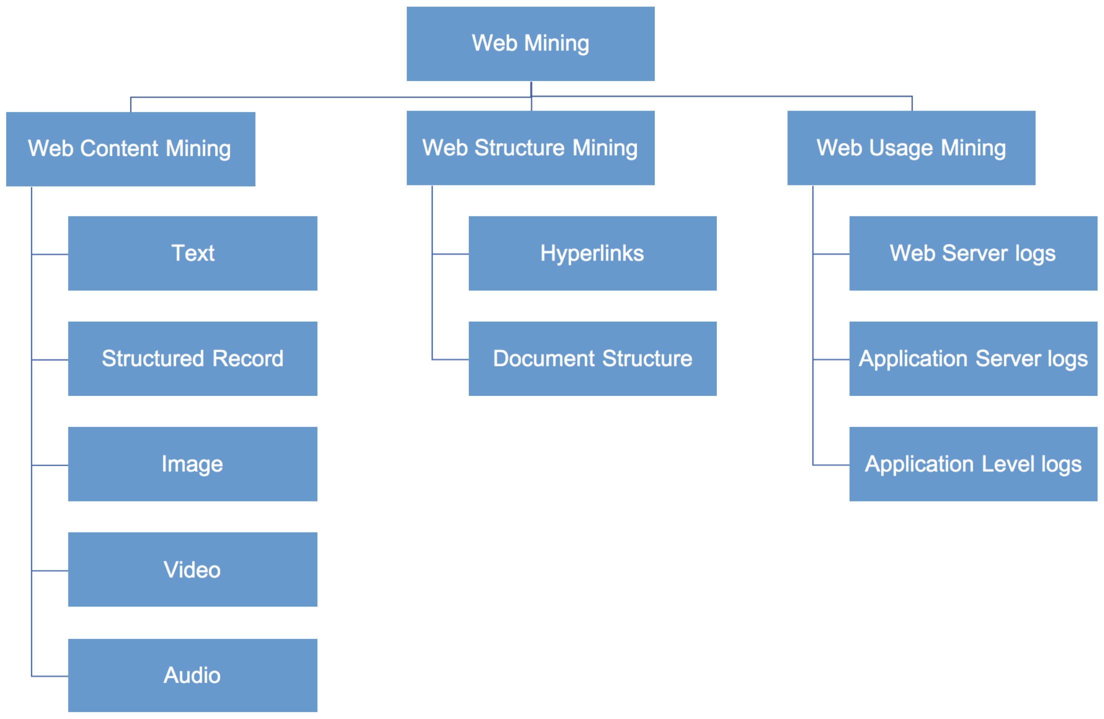


Figure 2.1: Categories of Web mining (based on [52])

- **Web Content Mining**

  Web content mining describes the process of discovering useful data or knowledge by extracting or mining information from websites or Web website documents. This process is also called Web text mining, since text is the most widely analyzed content type on the Web [5]. It focuses on two different aspects: Information Retrieval (IR) and Database (DB) [11]. While the goal of IR is to assist or enhance the finding or filtering of useful information based on user profiles, the DB point of view tries integrate the data from the Web in a model so further queries can be performed.

  Content data can be seen as collection of facts, which are designed to be contained by websites. So websites may contain content in the form of (unstructured) text, structured records (e.g. lists and tables), images, video or audio [52]. Usually IR and NLP is used to gather these useful content data [5].

- **Web Structure Mining**

  Web structure mining can target two slightly different purposes. On the one hand, it aims to analyze the underlying hyperlink (short link) structure of the Web to discover a structural summary in the form of a model. The focus lies on an important aspect of Web mining: link information. The gathered model can be used to explore the relationship and similarities between websites [33]. Furthermore, a kind of trust level can be derived from Web structure mining, since websites that are linked to by a lot of other websites are usually offering high quality or at least interesting content [39].

  On the other hand, Web structure mining can also try to discover the structure of the Web documents themselves and is then called Web document structure mining [41]. Here, the goal is usually to extract Document Object Models (DOMs) to gather useful information based on the tree structure format of Hypertext Markup Language (HTML) and Extensible Markup Language (XML) documents [52].

- **Web Usage Mining**

  Web usage mining is the process of collecting and analyzing the data generated by the website's users. As a result of the users browsing activities, data such as cookies, mouse clicks, browser logs, registration data, sessions, Web server access logs and so forth can be automatically generated and stored by Web servers to perform further analysis such as discover activity patterns, measure the website's management and the user behavior. Besides that, Bing Lui mentions in [39] the issue of pre-processing the correct click stream data to collect the right data for the later mining. Kosala und Blockeel distinguish Web usage mining from Web structure and content mining as follows:

  "While the Web content and structure mining utilize the real or primary data on the Web, Web usage mining mines the secondary data derived from the interaction of the users while interacting with the Web." [33, p. 4]

Since the empirical part of this thesis researches on the content extraction from heterogeneous online sources, the following sections focus more on Web content mining.

2.1.3 Subtasks of Web Mining

There are many tasks which are related to Web mining, such as ... TODO: Quelle. The following four can be seen as the summarizing and general tasks [33, 19]:

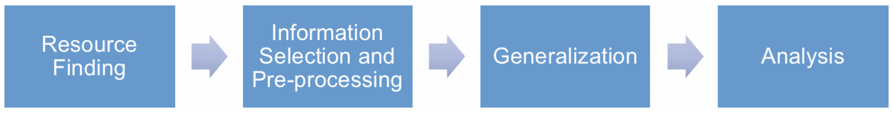


Figure 2.2: Subtasks of Web Mining (based on [33])

1. **Resource Finding**

   In a first step, the data must be retrieved from antecedently selected online sources such as RSS feeds, email newsletters and content from HTML websites (e.g. blogs, wikis, ...). This involves the tasks of identifying useful information, which requires tools for navigating and searching on websites, as well as reading and indexing to enable further analysis. This can be accomplished by utilizing Web crawlers.

2. **Information Selection and Pre-processing**

   In this step, information gathered via the retrieving process is selected and pre-processed automatically which means that some kind of transformation take place. These transformations could be pre-processing tasks such as removing stop words[1], stemming and identifying phrases in the training corpus to match the desired data representation. This task is usually performed by content parsers and adaptive wrappers.

3. **Generalization**

   Typically machine learning or data mining is used to discover general patterns in the extracted data.

4. **Analysis**

   In the last step, the mined patterns are further assessed to achieve a validation and interpretation.

[1]Classification of words can be an important tasks in several applications of data mining. Therefore, a removal of the most common words (called stop words), such as "the" and "and", often takes place to facilitate the process of identifying the significant parts of documents, since these word do not contribute useful meaning to the topic. [35]

2.1.4 Challenges of Web Content Mining

The Web offers a nearly unlimited amount of data/information in various types and it is still growing. This information is created not by one, but millions of people and machines around the world, which leads to heterogeneity and thereby problems for Web mining on different levels [39]:

- **Redundancy of Web Content**

  Due to the diverse authorship and the sheer amount of Web pages, the Web is a place where information is often available redundantly. This means that the same or similar piece of information may appear in multiple pages, but differs in the way of expression or wording [12]:

  - **Same content - different expression**
    As already indicated above, the Web offers many ways of expressing content, such as tables, unstructured or semi structured text, multimedia data (e.g. pictures, videos, audio files, Graphics Interchange Formats (GIFs)) and so forth [39]. Taking into account this representation heterogeneity of content (especially multimedia data) can be a tough challenge during various Web mining tasks [19].
  - **Same content - different wording**
    The next issue is closely linked to the previous point. Due to the diverse authorship of content published on the Web, same or similar information can be worded in completely different ways. This poses problems, especially when integrating information from multiple websites in a data model [39]. The utilization of NLP (see Chapter 3) can remedy the situation to interpret the context of text by a machine.

- **Contrariness of Web Content**

  Associated with the huge amount of different authorships, multiple websites can provide information which is slightly or completely contrary to each other. In this case, a fully automated decision on which sources provide the truthful content can be a difficult task. [19]

- **Impermanence and dynamic of Web Content**

  Since the Web is a high dynamic place, the information on websites changes constantly and a guarantee of permanence in terms of content storage is not given (*"sites appear and disappear"* [12, p. 1]), monitoring and keeping up with changes can be an important issue for various applications of Web mining. [39, 12]

In summary, it can be said that the heterogeneity, redundancy and the contrariness of content from online sources is caused by the diversity of authorship. These characteristics make the gathering of useful information from online sources - the overall goal of Web mining - a tough task. [39, 19].

2.2 Identification of trustable Online Sources

The identification and selection of credible online sources are crucial steps to gain meaningful and valid information through a subsequent Web mining (see Section 2.1). An opinion about the credibility of a website can be formed either manually by a Web user or automatically by a machine (e.g. search engines). Since the selection of suitable online sources for the empirical part of this work will be done manually, we just give a brief overview of the automated evaluation of web pages and will then discuss various quality factors for online information. The collected information about how the credibility of online information can be measures, is intended to help in the selection of suitable online sources. These sources serve as data bases for the prediction of upcoming technology updates.

2.2.1 Assessment by Search Engines

This chapter is intended to provide an overview of how search engines rank their results. This is deliberately kept short as the search for suitable sources of information for the empirical part of this thesis will be done manually.

The main goal of search engines is to find the best web pages that match a search query and rank them using different factors [40]. There are many search engines worldwide, but just four share over 98% of the market (see Figure 2.3). Google[2] is clear market leader with 79%. Bing[3], a search engine developed by Microsoft, and Baidu[4], a search engine developed for the Chinese market, each possess about 7% market share. The formerly very popular search engine Yahoo[5] currently has only a market share of about 4% [46]).

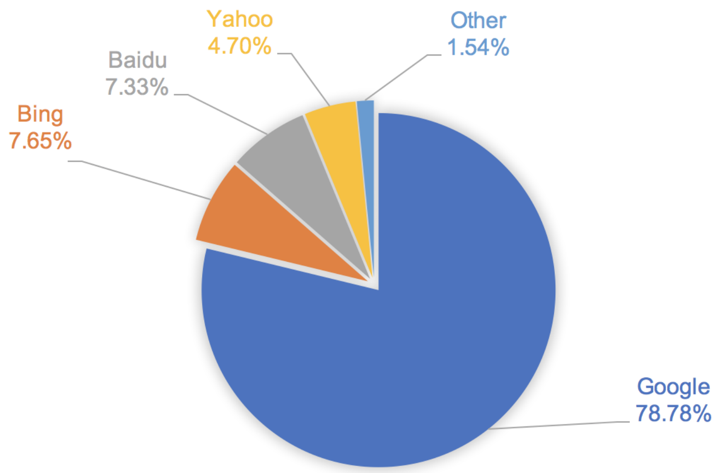


Figure 2.3: Desktop Search Engine Market Share (Data from [46])

[2]https://www.google.com
[3]https://www.bing.com/
[4]http://www.baidu.com/
[5]https://www.yahoo.com/

Search Engines aim to find the best results suited to the user's needs. Therefore, a ranking algorithm has to be implemented, which is to provide matching search results on the basis of various factors. These algorithms are based on a retrieval model and are often not accessible to the public. Part of this retrieval model is usually some kind of link analysis. Links are not only an important part of the Web, since they connect Web pages, but also help search engines to understand relationships between websites. [13]

Larry Page and Sergey Brin [6] laid the foundations for the analysis of links and their quality when they developed the algorithm PageRank at Standford University. Their algorithm measures the popularity of websites by not only determining links that are leading to it, but by considering the popularity of the linking websites as well [32]. However, according to Matt Cutts [14], Google's ranking algorithm comprises nowadays over 200 signals, and PageRank is just one of them. He divided these signals broadly into two categories:

- *Trust* is an assessment on a web page's authority and reputation.
- *Relevance* is an assessment on how well a web page matches a particular query and how topical it is.

He further mentions that the challenge on finding the best results is on the one hand to show web pages with high reputation and high level of trust, and on the other hand the web page has to offer the content in regard to the query the user typed in. [14]

In summary, it can be said that a high ranking of a web page may be an indication that it provides qualitative-appealing content to a given search criteria.

2.2.2 Assessment by Web Users

Various factors can be considered when a user forms an opinion about a web page and its content. In the previous section, we gave a short overview of how search engines assess the quality and credibility of websites. In return, this section gives an overview of various criteria for measuring the quality of websites manually that have been tested in the course of studies or suggested by previous research activities.

The Technology Acceptance Model (TAM) from Davis [15] is an often cited model which aims to not only predict the user acceptance of information systems, but explain why a system may be unacceptable. The model is based on two particular beliefs:

- *Perceived usefulness* measures the degree, to which a user can increase their performance by using a specific information system.
- *Perceived ease of use* measures the degree, to which a user can operate the system without problems.

Jeong and Lambert [29] have tested a framework evaluating the content quality on lodging websites based on the TAM. Their study suggested that among perceived ease of use and perceived accessibility, the perceived usefulness and attitudes appeared to be the most meaningful indicators.

Cao et. al. [7] have developed a conceptional framework for measuring the quality of e-commerce websites, which is based on the TAM as well (see Figure 2.4). They measure quality of a website by system quality, service quality, information quality and attractiveness. However, they admit that this also depends to the website (e.g. social media, news, web shop, . . .) itself.

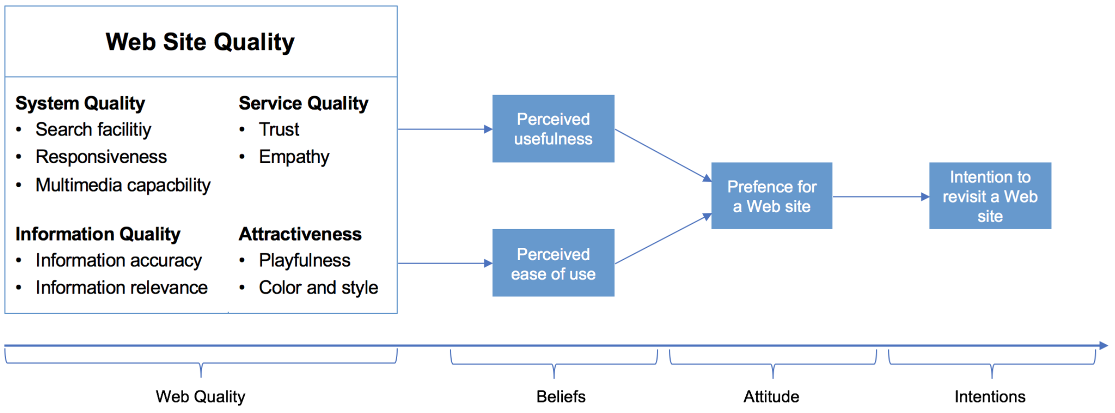


Figure 2.4: Framework for evaluating Website Quality (based on [7, p. 648])

Johnson and Misic [30] introduced a metric for benchmarking various websites of schools and professional organizations. The benchmarking attributes are divided into three categories: (1) *functional/navigation issues*, (2) *content and style*, and (3) *contact information*. Among the first category, they mentioned factors, such as ease of finding the website, navigating throughout the website, ease of returning to the main page and loading speed. Date of last update, the effective use of color and graphics, the consistency in color and style and wording are important factors of the second category. The third category comprises factors, such as offering information about the websites maintainers, authors of the content and contact information (i.e. phone number and E-mail address).

Mich et. al. [44] developed a model to evaluate the website quality as well. In their work, the following categories are relevant in regard to the empirical part of this work:

- The category *Content* should evaluate the value of information and links as well as the quality of information, its sources and authors.
- The category *Location* observes factors, such as using intuitive Uniform Resource Locators (URLs), providing contact information and forcing community building.

- The category *Management* should ensure that the information is up-to-date and the latest change date is evident.
- The category *Usability* comprises attributes in regard to the accessibility (i.e. hardware and software requirements and people with disabilities) and navigability (i.e. the websites structure and download speed).

Ethier et. al. [16] categorize the research on website quality in various groups. One of them focuses on *"functional and navigational issues (speed and ease of navigation), content and style (currency and presentation), and contact information"* [16, p. 628], while another focuses on system, information and service quality.

Wathen and Burkell [59] examined the process of how users form an opinion on the credibility of information on web pages (see Figure 2.5). They proposed a model which represents the judgment on credibility as an iterative process. After entering a website, the user makes immediate judgment about the appearance, interface design, download speed, organization of the website, and so on and questions like *Does this web page make a professional impression?* and *Is the information quickly and easily accessible?* arise. When these criteria are satisfied, the user moves to the next evaluation level. Otherwise, he or she will probably leave the page, but this can differ among users. On the second level of evaluation, the provided information and thereby factors such as trustworthiness, currency, competence, and level of detail are considered to assess the website's credibility. Again, questions arise, such as *Is the website providing the information I am looking for?*, *Do I believe the presented information?* and *Can I tailor the information to my situation?*. In a last step, the content evaluation take place which includes task such as comparing the information to previous knowledge.

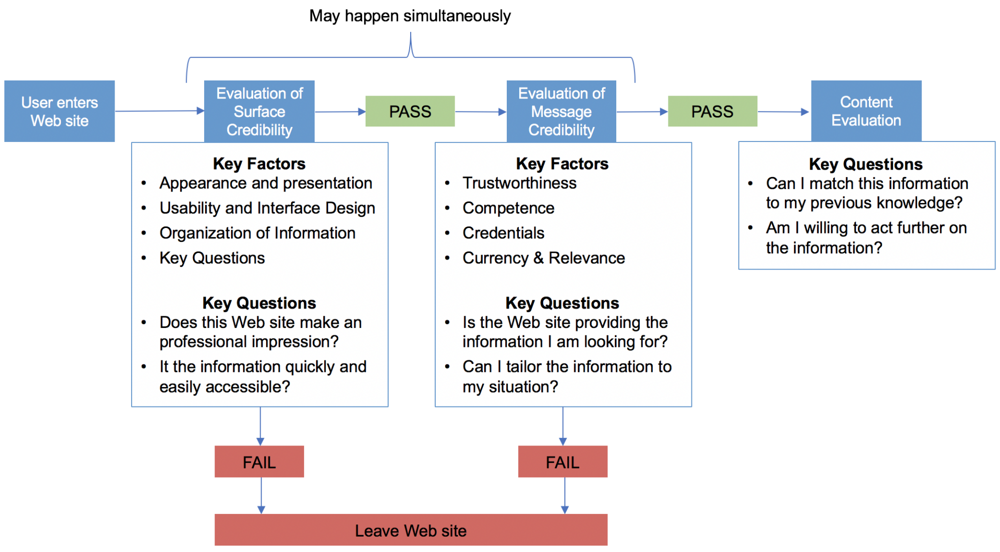

May happen simultaneously
User enters Web site
Evaluation of Surface Credibility
PASS
Evaluation of Message Credibility
PASS
Content Evaluation
Key Factors
• Appearance and presentation
• Usability and Interface Design
• Organization of Information
• Key Questions
Key Questions
• Does this Web site make an professional impression?
• It the information quickly and easily accessible?
Key Factors
• Trustworthiness
• Competence
• Credentials
• Currency & Relevance
Key Questions
• Is the Web site providing the information I am looking for?
• Can I tailor the information to my situation?
Key Questions
• Can I match this information to my previous knowledge?
• Am I willing to act further on the information?
FAIL
FAIL
Leave Web site

Figure 2.5: Model of how User judge the Credibility of Websites (based on [59, p. 144])

Metzger summarizes in [43] the skills users need to judge on the credibility of online information, which is also confirmed by up to date academic websites (e.g. [47, 37, 36]). She described in her literature review the five quality criteria as follows: (1) *Accuracy* measures the reliability and the error rate of a website. (2) *Authority* refers to the people who authored the content, their credentials, and qualifications, and which person or organization is providing the website. Whether a website is recommended by a trusted source can also be included in the evaluation. (3) *Objectivity* refers to the task of determining the purpose of the website. In the course of this, the questions should be examined as to whether the content is a fact or an opinion and whether a commercial interest exists. (4) *Currency* should establish whether the provided content is up to date. And (5) *Coverage* refers to the task of verifying the comprehensiveness of the website's content.

After the analysis of previous work, the conclusion can be drawn that website quality is first, *"an unclear and complex concept that has multiple dimensions"* [16, p. 629] and second, a concept, which does not have a unified definition [1, 16]. Thus, the above mentioned criteria of the different authors cannot be considered as generally valid and applicable to all types of websites and applications, but should be adapted according to the website itself and the reason for the evaluation.

2.3 Search Engine Analysis

In Section 2.2.1, we have already introduced the various definitions and tasks of search engines. Furthermore, it was shown that Google dominates the global search engine market and Bing takes second place in that ranking [46]. In this section, we will first show that the analysis of search engines can help to predict certain events (see Section 2.3.1). In addition, since Google is - to other knowledge - the only search engine operator offering an analysis of the trends for the English-speaking region, we will investigated the analysis capabilities of Google Trends both manually (see Section 2.3.2) and automated (see Section 2.3.3).

2.3.1 Related Work

Already in 2005, papers were published on the topic of search engine analysis. For example, Ettredge et. al. showed in [17] the potential of tracking Web searches to predict the unemployment rate in the United States by the assumption that Web user reveal their concerns, needs, interest, and so on via their behavior on the Web.

The analysis of search engines was used not only for the prediction of economic statistics but was also relevant for the medical field. For example, Ginsberg et. al. [20] and Polgreen et. al. [48] have shown in the field of epidemiology that using Web searches may help to predict influenza-like illness in a population, since the relative frequency of certain search queries can correlate with physician visits where a influenza-like symptoms are diagnosed.

The financial sector has discovered the potential for analyzing search queries as well, as the paper of Preis et. al. [50] demonstrates. They have found patterns that may predict stock market movements based on the changes happen in Google query volumes.

The above examples have shown that by analyzing Web searches, in certain cases a model for prediction can be created. As far as we know, no paper has been published to show the capabilities of predicting technology updates by analyzing search engines. This open research question will be answered in the empirical part of this thesis. Since Google is the only search engine operator offering an analysis of the trends for the English-speaking region, the analyzing capabilities of Google are now further investigated.

2.3.2 Analyze Web Searches with Google Trends

As Google is, according to our knowledge, the only search engine operator which allows an analysis of Web searches for free, this chapter examines the capabilities of this analysis both directly on the Web and via Application Programming Interface (API).

Google provides an analysis tool that shows the trend towards different topics or search terms, called Google Trends[6]. More specific, the number of queries for a search term in a particular geographic region and time range is divided by the total number of search queries for the same region and time range. To achieve a meaningful result, the maximum number of queries in the specified time period is normalized to 100. [9, 22]

The UI is designed simple (see Figure 2.6) and works as follows:

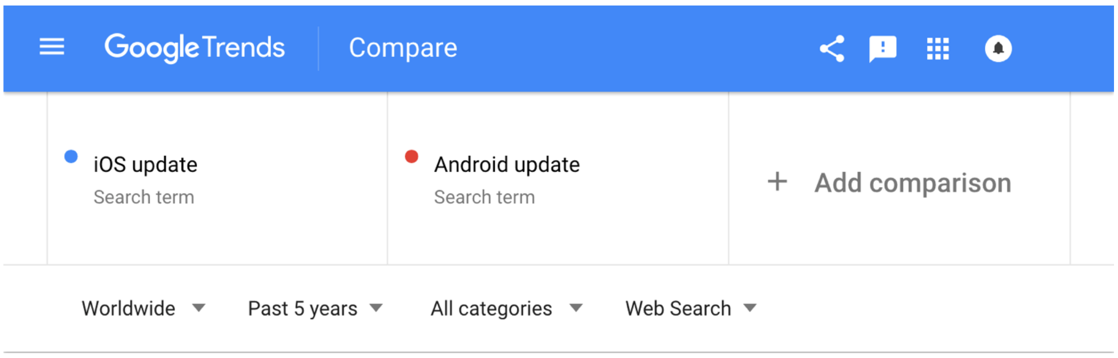

Google Trends
Compare
iOS update
Search term
Android update
Search term
+ Add comparison
Worldwide
Past 5 years
All categories
Web Search

Figure 2.6: Screenshot Google Trends Search Parameters

the user first enters a keyword, which he or she wants to have analyzed. Second, (1) the location (e.g. worldwide, Austria, Germany, New Zealand, . . .), (2) the time range, (3) the category (e.g. Autos & Vehicles, Computer & Electronics, Science, . . .), and (4) the medium (Web Search, Image Search, News Search, Google Shopping and YouTube Search) can be specified optionally. Furthermore, Google Trends enables a comparison of multiple keywords and combines the amount of Web searches in one graphic. For

[6]https://trends.google.com/trends/

example, if a user would like to compare the hype about the new software releases of iOS[7] and Android[8], he or she can do that as shown in Figure 2.6. The user enters the keywords “iOS update“ and “Android update“ into the designated fields and select a worldwide analyzation for the past five years for all categories with medium Web Search.

The output of Google Trend’s analysis can be seen in Figure 2.7. In the course of the analysis, we added the agenda, the dates on the abscissa, and the notes to the peaks of both graphs. The word “Note“ on the abscissa near Feb. 2016 is placed by Google and hints that they have applied improvements to their data collection system on 1.1.2016.

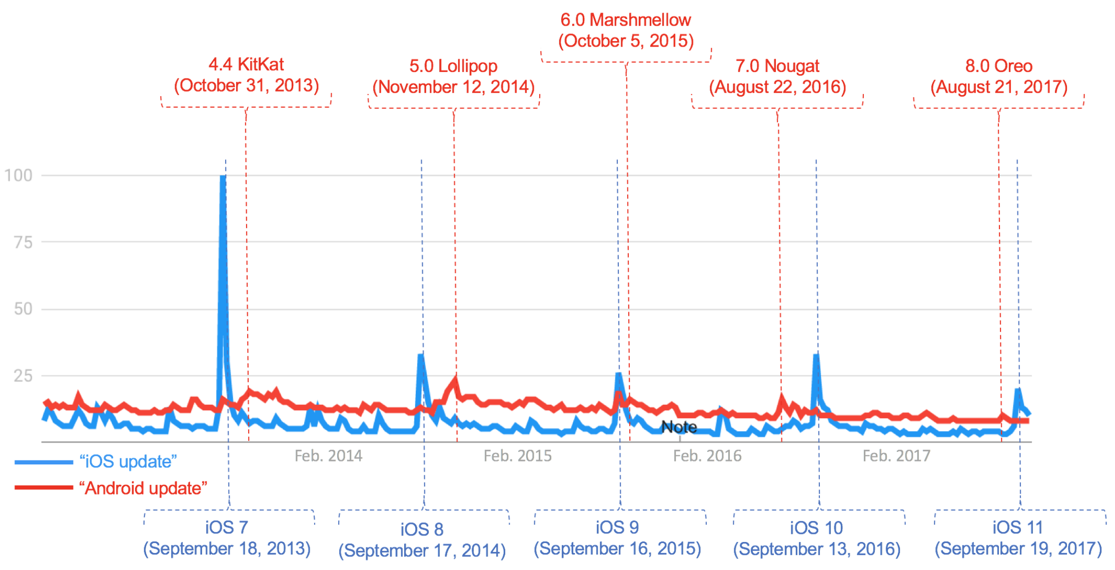


Figure 2.7: Modified Screenshot Google Trends

This graph shows the relative amount of searches for the keyword “iOS update“ (blue graph) and “Android update“ (red graph). Usually, Apple is releasing their software update for iOS in September, which correlates with the amount of search queries on Google (see periodic peaks in September of the blue graph). Android has a similar update cycle, which means that every year between August and November there is an increase in search queries as well(see peaks of the red graph). From the comparison of the two graphs it can be deduced that the interest in iOS updates at the time of the release is many times higher than that of Android. However, outside of the release time the interest in Android updates is higher.

In the blue graph, several small peaks can be seen which indicate an increased interest in an iOS update. This could be a reference to an increased occurrence of rumors, which in

[7]iOS is a mobile operating system, developed by Apple for their smartphones (iPhone) and tablets (iPads).

[8]Android is a mobile operating system developed by Google. It is usually used for touchscreen devices, such as smartphones and tablets and has often customized interfaces by various manufactures (e.g. Samsung TouchWiz, HTC Sense, Huawei EMUI, LG UX, Sony Experia UI, . . .).

turn can indicate an upcoming update. Thus, analyzing the trends in search engines can support the process of predicting upcoming technology updates.

2.3.3 Automated Access on Google Trends

In the section above, the usage of Google trends via its interface was demonstrated. This section provides information on how Google Trends can be accessed and used automatically without using the Web interface. Unfortunately, Google does not provide an official API[9], but the access via a GET request[10] is possible. If we would like to request the same example as above (searching for the terms "iOS Update" and "Android update" worldwide in the past 5 years for all categories in the medium Web Search, the following GET request is created:

```
www.trends.google.com/trends/explore?date=today%205-y&q=iOS%20
update,Android%20update
```

The GET request contains the parameter date with value *today 5-y* for the past 5 years and the parameter q with value *iOS update,Android update* for the search query. The parameter for the geographic region "Worldwide", the category "All categories" and the medium "Web Search" are not represented in the request, since these are default values. In contrast to this, the search for the keywords "WatchOS update" and "Android Wear update" in the past 90 days for the region "United Staes", the category "Computers & Electronics" and the medium "Youtube search" looks as follows:

```
www.trends.google.com/trends/explore?cat=5&date=today%203-m&geo=
US&gprop=youtube&q=WatchOS%20update,Android%20Wear%20update
```

In addition to the parameters described above, the parameter "cat" with value "5" for the category "Computers & Electronics" and the parameter "gprop" with value "youtube" for the medium were added.

Besides the GET request, a few unofficial APIs are available, all of which provide similar access to the functionality of Google Trends:

Patrick Trasborg has developed the *google-trends-api*[11], which provides an API layer to Google trend's data. It is part of the npm[12] project and therefore developed in JavaScript. It offers various methods, such as *interestOverTime* to retrieve the numbers of search queries relative to the highest point on the chart for the given geographic region and time period; *interrestByRegion* to get the regions where the search term was most popular for

[9]APIs enable other operating systems or applications to access functionality via a predefined request by providing an (documented) interface. [55]

[10]GET is one of the two Hypertext Transfer Protocol (HTTP) request methods (the other is POST) and is designed to request data from a specified resource (e.g. URL of a Web server). [58]

[11]`https://www.npmjs.com/package/google-trends-api`

[12]npm is a package manager for JavaScript and is used by the open-source JavaScript runtime environment NodeJS by default.

the given time period and *relatedQueries* to get the queries searched by Users who also searched for the defined query. [57]

John Hugue is providing a API for accessing Google Trend as well, called *pytrends*. The API is developed in Python and allows an automated download of reports. The main feature is the login script to Google. [23]

Marco Tizzoni has developed a Java based solution for accessing Google Trend, which provides classes to authenticate to Google's services, parsing the Character-separated Values (CSV) Files requested from Google Trends and supporting Proxy configuration to avoid Google's query limits. [56]

In summary, it can be said that the automated access to Google Trends is limited to a few unofficial APIs and the possibility that Google Trends accept GET requests with easily adjustable parameter structure. The empirical part of this work will show the extent to which these analytical possibilities contribute to the prediction of technology releases.

CHAPTER 3

Concepts of Text Analysis with NLP

In the previous chapter, we looked at various aspects of information retrieval from heterogeneous online sources and therefore have presented Web Mining as a central anchor point of information acquisition. We further investigated the process of how credible and relevant information sources can be assessed and how capable search engine analysis is for predicting certain events. All these points provide us raw data, which must be further processed under certain aspects in order to generate knowledge.

In this context, we investigate NLP, which is intended to help the processing of the extracted information and to further support the prediction of technology updates. Therefore, we first give an overview about common definitions and characteristics of NLP (see Section 3.1) found during the literature review and compare it against each other. In Section 3.2, we show the different level of analysis which can be performed by NLP tools and where they reach their limits. Various application areas of NLP are presented in Section 3.3. The last Section of this chapter investigates four major tools in linguistic analysis (see Section 3.4):

- Sentence delimiters and tokenzizers (3.4.1)
- Stemming/lemmatizing and tagging (3.4.2)
- Noun phrase and name recognizers (3.4.3)
- Parsers and grammars (3.4.4)

3.1 Definition and Characteristics

NLP aims to analyze and convert human language, such as English, German, or Greek, into a more formal representation (e.g. first order logic, parse trees, etc.). This should enable computers to perform specific language interpretation tasks easier. Kumar describes it further on a meta level as *"the scientific study of languages from computational perspective."* [34, p. 1]. James F. Allen focuses in his definition more on the NLP pipeline. He assumes a computer system, which takes text, spoken language or keyboard input as input to analyze, understand or produce language. This should be done to fulfil task, such as building a knowledge database, generating summaries, translating into another language or maintaining a dialog with a user [2].

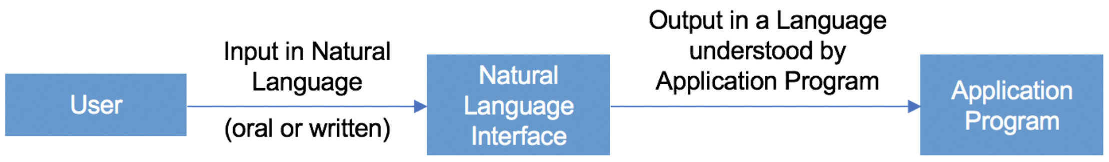

User
Input in Natural Language
(oral or written)
Natural Language Interface
Output in a Language understood by Application Program
Application Program

Figure 3.1: NLP System Pipeline (based on [34, p. 10])

An all-embracing definition of NLP including its various characteristics is provided by Elisabeth Liddy:

"Natural Language Processing is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications." [38, p. 2]

Based on the definitions introduced above, the following characteristics of NLP can be derived [38, p. 2]:

- The text phrase *"range of computational techniques"* is intended to show that there are various methods and techniques for analyzing language.
- By *"naturally occurring texts"* is meant that the text must be spoken or written and is formulated in a language, understandable to people.
- NLP can be performed on different levels of language processing (see Section 3.2), which is revealed by *"levels of linguistic analysis"*.
- Since human performance is desired with NLP (*"human-like language processing"*), it can be seen as a discipline of Artificial Intelligence (AI).

In addition to the characteristics of NLP, Kumar notes in [34] two different NLP systems: a *natural language generation system*, which is able to convert information stored in a database into human readable language and a *natural language understanding system*, which is capable of converting human language into a more formal representation interpretable by a machine. However, many NLP tasks require both generating and understanding.

3.2 Levels of NLP

The analysis of NLP is divided into several parts, such as phonological, morphological, lexical, syntactical, semantic, discourse and pragmatic analysis. The aim of this section is to introduce these analysis levels. Further, it is shown, which tasks can be performed by NLP systems easily and where they reach their limits. The following list comprises these tasks in more detail [34, 38]:

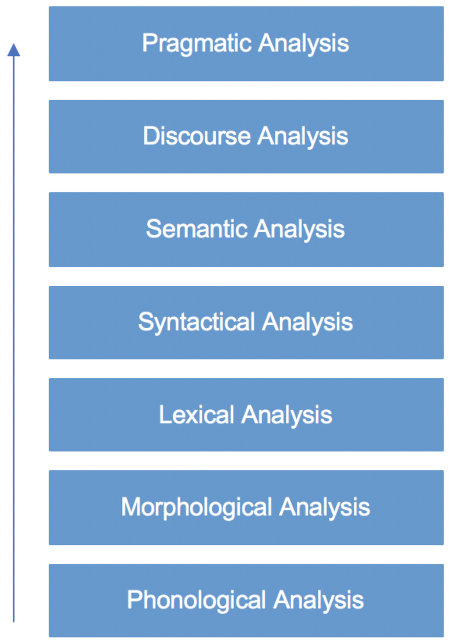


Figure 3.2: Levels of NLP

- **Phonological Analysis**

  Phonology is the science which deals with the analysis of the spoken language. In particular, acoustic waveforms are taken as input and after processing, a string of words is outputted. It is usually used for recognizing and generating speech. Since the prototype of this thesis is only analyzing text from the Web (and no audio), this level is not dealt with further.

- **Morphological Analysis**

  Morphology is one of the most important components of NLP and deals with the analysis of words. A word can be decomposed into its components, called morphemes. They represent the smallest meaningful units of a language elements and if it forms the essential part of the word, it is also called *stem*. In English, for example, a morphological analysis might look as follows:

Word: Search	–> stem “search“
Word: Magnetize	–> stem “magnet“, suffix “ize“
Word: Depersonalize	–> prefix “de“, stem “person“, suffix “al“, “ize“

  Table 3.1: Morphological Analysis of Words (based on [34, p. 17])

  This morphological analysis allows people to divide unknown words into their parts in order to understand them. An NLP system makes use of exactly the same process to gain meaning about a word (e.g. adding the suffix *-ed* to a verb, indicates that the action has taken place in the past).

- **Lexical Analysis**

  In this stage, validity of words is checked with the help of a lexicon/dictionary, which is a collection of all valid words of a language. The process is as follows: Each word of a sentence is scanned and analyzed by looking up at a predefined dictionary. After the word was found in the dictionary, all relevant linguistic information is gained, such as the type of the word. For example, a lexical analysis of the word “run“ may point out that it is a verb.

- **Syntactical Analysis**

  In this phase, the sentences underlying grammatical structure is analyzed to unveil the relationship between the containing words. For this process, both a grammar and a parser are required. The grammar provides information about the formal requirements of a language, while the parses analyzes the sentence based on this grammar. Since the order and dependency of word influence the meaning of a sentence in most languages, the syntax contributes to understand a sentence. An example on how a parser works is presented in Section 3.4.4. According to Ela Kumar, Context Free Grammar (CFG) is the most common choice for syntactical analysis and is also called *phrase structure grammar* or *definite clause grammar* [34].

  CFG is a formal grammar and consists of *non-terminals*, *terminals*, *a starting symbol* and a set of production rules, which dictate how a valid string should look in the specified language. Non-terminals are symbols, which can be replaced by non-terminal as well as terminal symbols with the help of the grammar’s production rules. Although they can occur on both left and right-hand side of production rules, they are not part of the resulting formal language and thereby must be replaced by terminal symbols. Terminal symbols are elementary elements of a formal language, which may replace non-terminal symbols by applying certain production rules and cannot be further replaced. [34]

- **Semantic Analysis**

  This phase of natural language analysis tries to determine the possible meanings of a sentence. This is done by not only considering the word itself but its meaning in regard to the sentence and thereby the corpus. For example, the noun “nail“ describes both a part of a finger and a sharp piece of a particular material (e.g. metal) which can be used for construction purposes. If information from the rest of the sentence is required to clearly identify the meaning of a word, the semantic and not the syntactic analysis must be used. There are different approaches to accomplish a semantic analysis, as mentioned by Liddy:

  “A wide range of methods can be implemented to accomplish the disambiguation, some which require information as to the frequency with which each sense occurs in a particular corpus of interest, or in general usage, some which require consideration of the local context, and others which utilize pragmatic knowledge of the domain of the document.“ [38, p. 8-9]

- **Discourse Analysis**

  The aim of discourse analysis is to understand the meaning of the author by not only looking at a single sentence but understanding the properties of a text. This should form connections between sentences and, for example, enable the system to match pronouns to the entities they refer. For example, the sentence “The teacher points at the student to let her answer the question“ suggests that the student has raised her arm in order to answer the teacher’s question. Understanding this context can be a tough tasks for NLP systems.

- **Pragmatic Analysis**

  In this phase, a relationship between language and context of use is established. Therefore, a NLP system must be capable of identifying references to people and things, which is a complex task and required pragmatic or world knowledge. For example, in the following two sentences, it is necessary to ask a kind of knowledge base to determine the correct referencing of the word *they* (based on [38, p. 9]):

  Two teachers honour a group of pupils as they appreciate their performance

  Two teachers honour a group of pupils as they performed outstanding at a contest.

  These two examples are intended to show that the understanding of a sentence can be a complex task for a system, although it is usually easy for a human being. [38]

3.3 Application Area

Since NLP deals with the processing of natural language in written or oral form, there is a very broad and cross-sectoral application area, such as machine translation, Cross-Lingual Information Retrieval (CLIR), AI, and speech recognition [10]. However, the application areas of NLP can be categorized according to the following characteristics [27]:

Reproduce vs. transform. NLP systems of the category *Reproduction* are mainly used for reproducing or reinstating linguistic phenomenon, whereas systems of the category *Transformation* are specialized in the translation of texts into another representation form.

Recognize vs. generate. NLP system can also be divided into recognizing and generating applications. On the one hand, *Recognition* unites application areas, which focus on recognizing or analyzing linguistic inputs. On the other hand, systems of the category *Generation* are concerned with generating or synthesize linguistic output.

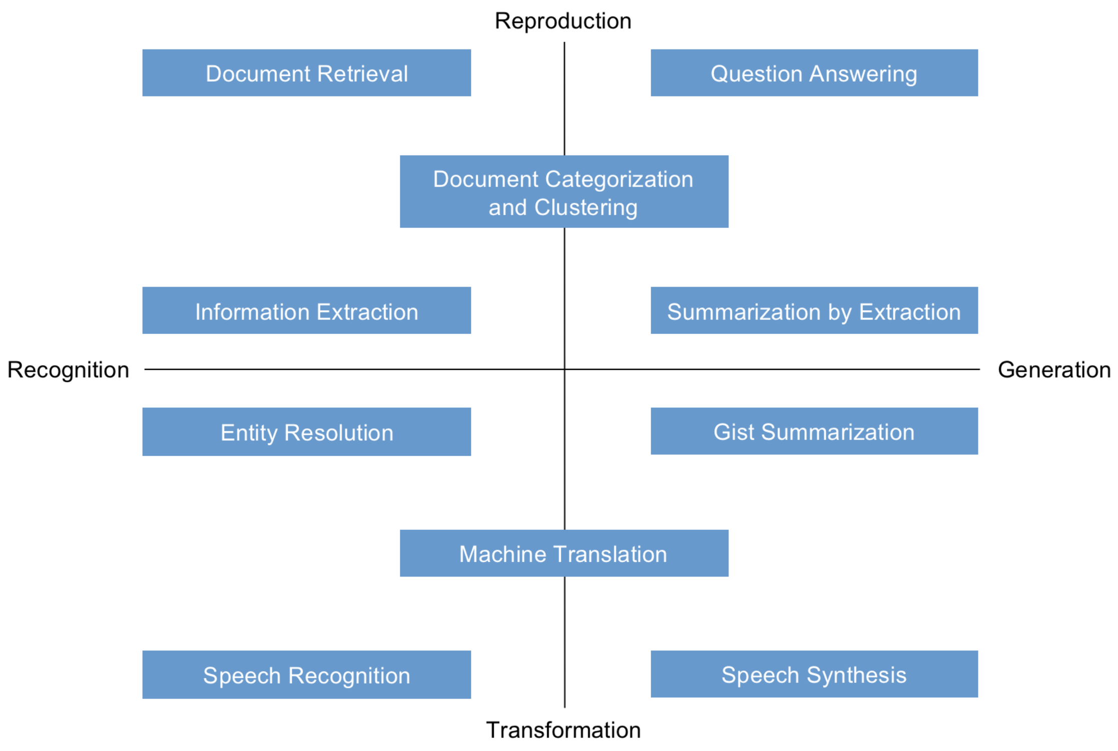


Figure 3.3: NLP Applications (based on [28])

However, applications are not always attributable to just one of the four dimensions. For example, while *machine translation* first performs recognition in the source language to create the base for generating it into the target language, *document categorization and* *clustering* requires first a recognition of content types before generating the clusters of categories. [27]

Thus, applications can also be a mixture of two or more categories. In line with the above-described categorizations of the application areas of NLP, a graphic is given, which lists examples for the respective categories:

For example, the aim of Information Extraction (IE) is to extract certain elements of information, such as names, locations, dates and so forth, by finding, tagging and extracting these elements in a text. Another example is Question Answering systems, which provides appropriate answers to user queries. These answers are selected by processing the user queries. A higher-value application is the summarization, where NLP systems try to short a text automatically without losing substantial content. [38]

3.4 Tools in Linguistic Analysis

In Section 3.2, we already discussed the possible phases in which a NLP system can operate. In this section, various methods are presented together with examples to explain the different functions of linguistic analysis in more detail.

Linguistic analysis is one of the basic instruments of NLP systems and is usually proceeded in layered fashion. This means that a document is first subdivided into paragraphs, the paragraphs subsequently into sentences and these sentences into individual words, which are then further processed according to the application [27]. The following tools can be used to enable the above-described processing of text:

3.4.1 Sentence Delimiters and Tokenizers

Before the analysis of texts can begin, they have to be divided into sentences. Detecting the beginning and the end of a sentence is the task of a *Sentence delimiter*. The identification of a sentence can be more difficult than initially assumed, since the symbol for the end of a sentence “.“ can also be used for other purposes, such as “M.Sc.“, “Mag.“, “Dr.“ and so on. Further, a sentence can also be ended by “?“, “!“. The same applies to the beginning of sentences, since words which are not at the beginning of a sentence can be capitalized as well. Therefore, various mechanisms can be applied to make a correct decision, such as regular expressions[1]. [27, 3]

Tokenizers have a similar task to sentence delimiters. They are used to decompose a stream of characters into meaningful units called token. Again, the task of a tokenizer seems to be simple, since only a separation by spaces must take place. However, this does not apply to all languages (e.g. words in Chinese or Japanese are note separated by spaces). Furthermore, tokenizers may have to fulfil even more complex tasks. For example, processing dates (e.g. “2017-09-29 13:00“ vs. “13h 29.9.2017“), monetary amounts (e.g. “€10.33“ vs. “10.33“), or composed words, as they are used in German

[1]Regular Expression are used to identify specific strings in a text by using a pattern. [8]

(e.g. “ Straßenverkehrsordnung“), may represent a difficult task [27]. However, the processing of noun compounds (e.g. people’s names or city names), can be an even more complex task. For example, the geographic locations “New York“, “New Zealand“ or “St.Veit an der Glan“, are difficult to process due to the division by spaces and points. Improvements can be achieved by applying name recognizers (see Section 3.4.3). [3]

3.4.2 Stemming/Lemmatizing and Tagging

Stemmers are used to perform a morphological analysis of a word, which is described under Section 3.2. The goal at stemming is to find the root form of a word (e.g. *grow* is the root form of grew, grown, growing, and - of course - grow). This can be done by either considering a dictionary/lexicon to look up for a word’s stem which is somewhat expensive, or applying a heuristic stemmer, which removes certain pre- or suffixes to discover the root form (e.g. removing “-ed“,“-ing“,“-ness“). The first variant is commonly referred to as lemmatizing, the second one to as stemming. Furthermore, stemmer could be used to decompose compounded terms. For example, “Straßenverkehrsordnung“ might be stemmed into “ Straße#Verkehr#Ordnung“, so the parts of the compounds can be further analyzed. [27, 42, 3]

Part-of-speech (POS) taggers are responsible for the assignment of suitable tags (e.g. noun, verb, or adjective) to words and are therefore based on tokenizers and sentence delimiters. For example, if we would like to tag the sentence “Visiting grand parents can be exhausting“, we are not able to tag the word “Visiting“ correctly, since the given information from the example is not sufficient (it can be tagged either as adjective or verb). In order to tag the word properly, the context of the word has to be considered. This can be done by either considering some rules, such as *“If an unknown term is preceded by a determiner and followed by a noun, it is to be marked as a verb“* [27, p. 15], or following a stochastic approach, which relies on training with a data set and is then based on frequency information or probabilities. [27, 3]

3.4.3 Noun Phrase and Name Recognizers

The main goal of noun phrase extractors is to identify base noun phrases and - if available - its left modifiers (i.e. adjectives and delimiters on the left side of the noun). Name finders go one step further and try to assign the identified nouns to a class, such as places, companies, people, and so forth [27]. Applying such a name finder to the sentence,

“The Sony Centre from Berlin was bought by the real estate company Centre Oxford Properties and the investment company Madison International Realty for 1.1 billion dollars last month.“

may lead to the following tagging:

Word	**Tag**
Sony Center	Place
Berlin	Place
Centre Oxford Properties	Company
Madison International Realty	Company
1.1 billion dollars	Amount of money
last month	Date

Table 3.2: Tagging Words by a Name Finder (based on [27, p. 16])

The sentence offers a challenge for the name finder, since the word “Oxford“ of the company name “Centre Oxford Properties“ also represents a city in England. In order to avoid groping in this case, a name finder must eliminate some degree of intelligence. [27]

3.4.4 Parsers and Grammars

The task of parsers is to scan a sentence and make checks in regard to syntactical and semantic analysis. This is done by the help of a grammar, which basically defines how well-formed sentence are structured by parts of speech [27]. For example, a sentence’s structure can be shown in a tree representation:

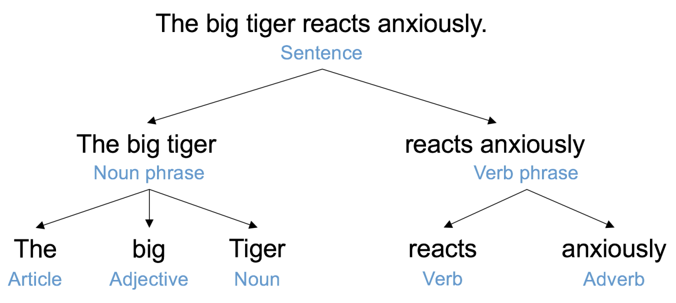


Figure 3.4: Example of a parser application (based on [34, 28])

The sentence is first divided into noun and verb phrase. These two phrases can be further split into article, adjective and noun for the noun phrase and verb and adverb for the verb phrase. This analysis can be helpful in determining whether a sentence corresponds to the underlying grammatical rules. Furthermore, semantic analysis can also be proceeded in order to learn more about the content. [34, 28]

CHAPTER 4

Design of Models for detecting Technology Updates

In order to answer the research questions of this thesis sufficiently, we need on the one hand a data pool and on the other hand analysis approaches to transform these data into knowledge. To collect and process the data, we needed to develop a prototype (see Section 4. This prototype is based on three generic models, which are introduced in the following sections.

First, we give an overview about how the proposed process of predicting technology updates looks like by introducing the *Process Model for Technology Update Detection* (see Section 4.1). Afterwards, we show how heterogeneous data sources are categorized in the *Categorization Model for Online Data Sources* (see Section 4.2) to collect information about technologies. This information is processed by the *Decision Model for Technology Update Notification* (see Section 4.3), which analyzes the previously extracted data and checks, whether enough information about an upcoming technology release was found.

4.1 Process Model for Technology Update Detection

In this section, we sketch the developed process of finding information about upcoming technology releases. Therefore, we describe the various subtasks in detail to give an understanding about how complex each step is and where a domain expert is still necessary.

Figure 4.1 gives an overview about the release prediction process, which can roughly be divided into two main tasks. First, the domain expert has to set up the system by (1) defining a technology and associated keywords, (2) identifying trustable online sources and developing data extraction methods, and (3) configuring the data export. The second main task consists of data mining and analysis which is done by the system.

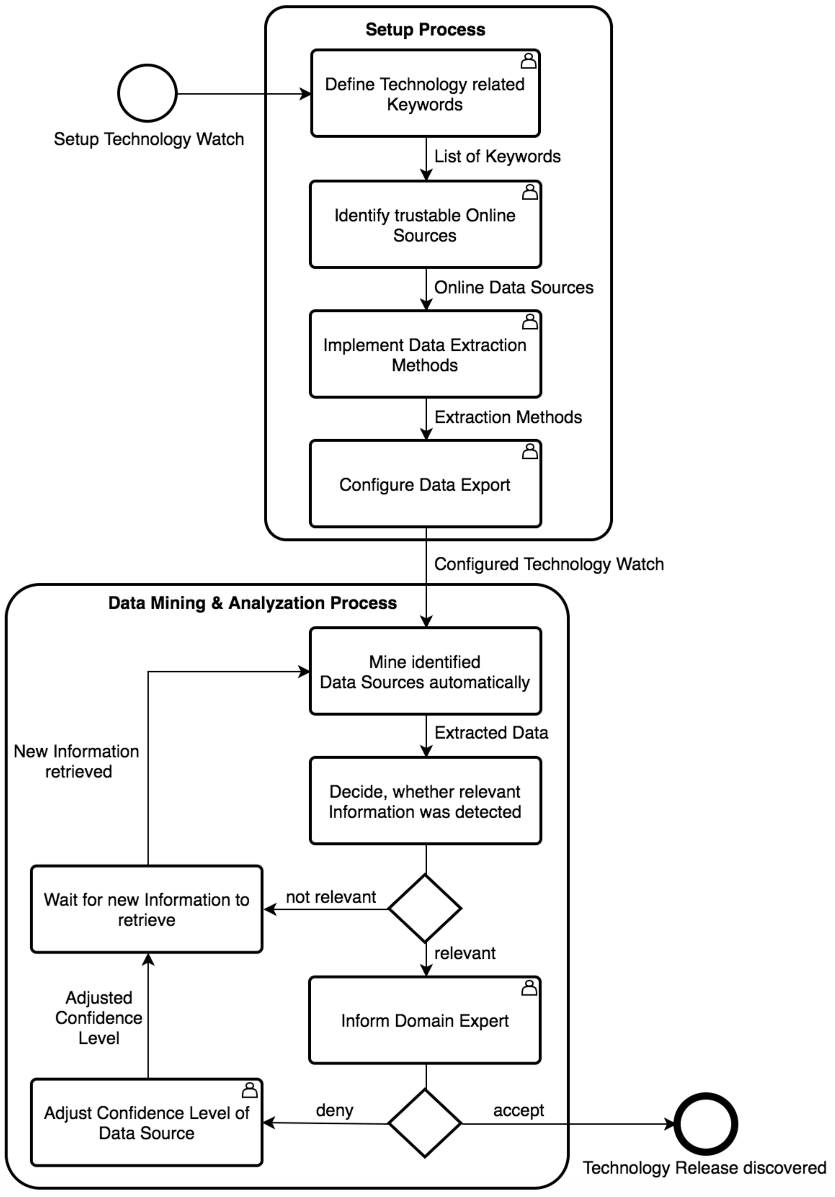

Setup Process
Setup Technology Watch
Define Technology related Keywords
List of Keywords
Identify trustable Online Sources
Online Data Sources
Implement Data Extraction Methods
Extraction Methods
Configure Data Export
Configured Technology Watch
Data Mining & Analyzation Process
Mine identified Data Sources automatically
New Information retrieved
Extracted Data
Decide, whether relevant Information was detected
Wait for new Information to retrieve
not relevant
relevant
Adjusted Confidence Level
Inform Domain Expert
Adjust Confidence Level of Data Source
deny
accept
Technology Release discovered

Figure 4.1: Process Model for Technology Update Detection

The following list comprises the subtasks of the process described in more detail:

1. **Define Keywords**

   Before the search for suitable online sources can start, the domain experts has to create a keyword list (*Define Technology related Keywords*). These keywords are used in the next step to discover appropriate online sources. For example, if the system should predict an upcoming release of a new Microsoft Windows or Apple iOS version, the keywords might be defined as follows:

Technology	Microsoft Windows	Apple iOS
Keywords	Microsoft Windows	Apple iOS
	Windows	iOS
	Windows 11	iOS 12
	Win	iPhone

   Table 4.1: Example of Keyword List

   However, the focus during the selection of suitable keywords should be placed on the quality of keywords rather than the quantity.

2. **Identify trustable Online Sources**

   After the domain expert has defined the keywords, he or she has to search for trustable online sources, which are later harvest to receive release information. The selection process should consider quality and credibility aspects, which were introduced in Section 2.2. The output of this step is a list of selected online data sources.

3. **Implement Data Extraction Methods**

   For each data source, the domain expert or a programmer has to develop the data extraction methods in order to gain data automatically. The complexity of this task highly depends on the selected data sources.

4. **Configure Data Export**

   In the last step of the setup process, the domain expert has to configure the various aspects of the system, including how often the system should extract data. After this step, the system is configured properly and is ready to gain data.

5. **Mine identified Data Sources automatically**

   After the setup process is finished, the system uses the developed data extraction methods to gain information from the various online sources. Again, the duration of this task depends on the selected data sources but should be much faster than performing the data extraction manually. The output of this task is the extracted information saved in the database.

6. **Decide, whether relevant Information was detected**

   In this step, the system analyzes the extracted data by using the Decision Model for Technology Update Notification (see Section 4.3). The four inputs - (1) extracted text phrases, (2) number of search requests (3) number of search results and (4) release information from online encyclopedias - are building the basis for calculating the Release Prediction Indicator.

   Before the calculation of the indicator starts, the system checks for updates on the online encyclopedia pages and RSS feeds/Twitter accounts, which are labeled as release channel. If one of these sources indicates a new software release, the system informs the domain expert directly. If that is not the case, the decision model processes the obtained data and decides, whether the extracted information contains a relevant indication for an upcoming technology release. Is that the case, again the system informs the domain expert (see step 8). If the extracted data does not indicate an upcoming technology release, the system has to obtain more information from other online sources (see step 7).

7. **Wait for new Information**

   If the calculated indicator of the previous step has shown that there is still no indication of an (upcoming) technology release, the system waits until new information is provided by the online sources. If new information is available, the system continues at step 5 by extracting data.

8. **Inform Domain Expert**

   In contrast to the previous step, the system informs the domain expert about a possible technology update, if the calculation of the Release Prediction Indicator has passed a certain threshold. The domain expert has to decide, whether the information found actually indicates an upcoming technology release. If this is the case, the domain expert accepts the result and the process reaches its final stage - a new technology update was found (*New Technology Release discovered*). If the indicator incorrectly implied a technology release, the update warning gets rejected and the confidence level of the data source has to be adjusted (see next step).

9. **Adjust Confidence Level of Data Source**

   If the domain expert denied the update warning, the system adjusts the confidence level of the data source. In more detail, at the beginning of the data extraction process the system assigns each data source an initial confidence level. For trustable online sources, like online encyclopedias or RSS feeds, which are hosted from a manufacture and just posts about new releases, the confidence level is fixed. For other data sources, like social media platforms or technology news, the system assigns a midrange value and over time, the confidence level gets adjusted according to the feedback of the domain expert. On the one hand, if the information published by the data source is valid, the confidence level rises. On the other hand, if the data source published an incorrect information, the confidence level decreases. This mechanism should filter dubious and not trustworthy data sources to improve the overall accuracy of the information presented to the domain expert.

4.2 Categorization Model for Online Data Sources

This section gives an overview about the selected heterogeneous data sources from which data was gained from in order to find information about (upcoming) technology releases. This information builds the basis for an in depth data source analysis.

The mined online sources can roughly be divided into four categories: (1) *Technology News*, which includes conservative online sources, (2) *Social Media*, which represents a more dynamic environment, (3) *Online Encyclopedias*, which provide concrete release information and (4) *Search Engine*, which provides information about the number of search results as well as the number of search requests to find trends. Figure 4.2 is a visual representation of the Categorization Model. In the following list, each category is described in more details:

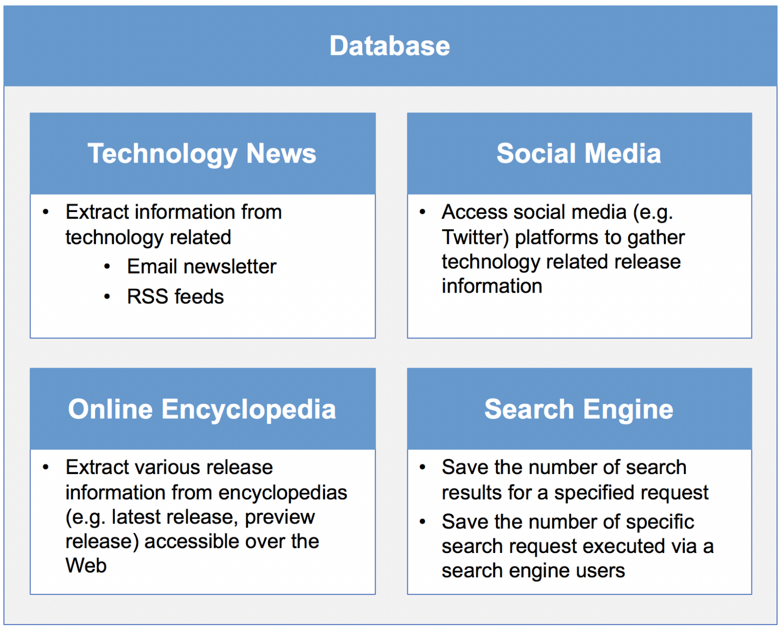

Database
Technology News
• Extract information from technology related
• Email newsletter
• RSS feeds
Social Media
• Access social media (e.g. Twitter) platforms to gather technology related release information
Online Encyclopedia
• Extract various release information from encyclopedias (e.g. latest release, preview release) accessible over the Web
Search Engine
• Save the number of search results for a specified request
• Save the number of specific search request executed via a search engine users

Figure 4.2: Categorization Model for Online Data Sources

Technology News. The Internet contains thousands of websites that report on technologies. We take advantage of this and mine selected Internet sources to obtain information of previously defined technologies. For the development of this prototype, we focus on two types of technology news sources: email newsletters and RSS feeds. Companies use email newsletters to inform their customers and business partners as well as interested persons about their activities, services and products. As email newsletters, RSS feeds are another method for spreading information.

Social Media. Since more and more companies refrain from sending newsletters, but relying on communication via social media platforms, we decided to extract information from Twitter[1]. Twitter is a social media platform founded in 2006 [26]. It allows its users to publish messages called Tweets, which are restricted to 280 characters[2]. To collect Tweets, we have created a Twitter account and subscribed to those Twitter accounts that eventually will publish release information about technologies chosen for the evaluation.

Online Encyclopedias. Our third category for retrieving release information is online encyclopedias. We decided to extract data from Wikipedia, since this platform offers release information uniformly. Thus, we do not have to implement individual data extraction methods for different data sources.

Search Engine. We have shown in Section 2.3 that with the help of analyzing search behaviour, a model of prediction can be created in certain cases. We further investigated the capabilities of Google Trends to analyze these behaviours (see Sections 2.3.3 and 2.3.2), since Google is - to our knowledge - the only search engine operator which provides such extensive analysis options. With the help of Google Trends and Google Search, we focused on two indicators:

- The first indicator is the *Number of Search Requests*. We obtain the total number of search requests for predefined keywords on a daily basis from Google Trends. An increase of keyword related search requests indicates that interest in the technology is rising. This can be due to various events, some of which may point to a technology update, such as a company announcing a product release, rumours about a technology update occured in the press, and so on.

- We further obtain the *Number of Search Results*, which represents the second search engine indicator. Assuming that the number of Web sites reporting an update may indicate an upcoming technology release, the total number of search results delivered from Google Search is obtained too.

4.3 Decision Model for Technology Update Notification

This section shed light on how the collected data from heterogeneous data sources are processed in order to find information about new releases and how the system decides, whether enough information was found to send a warning message to the domain expert. Before the process of calculating the release indicator can start, each configured data sources is checked for new information. To distinguish data sources based on their credibility and quality, each data source receives an initial confidence level. This confidence level may change over time. Changes are applied whenever information from a data source is presented to the domain expert and he or she decides whether the

[1] `https://twitter.com/`

[2] Before November 7, 2017, Tweets were restricted to 140 characters. [51]

information is valid. After the extraction of new information is done, the system is able to process the workflow of calculating the release indicator.

After the release indicator was calculated, the system decides, whether enough information about an (upcoming) technology release was collected to inform the domain expert. If this is the case, all findings for a specific technology are summarized in a warning message. This message will be sent to the domain expert and he or she decides, whether the warning was justifiable. If this is not the case, the system has to collect more data and then go through the analysis workflow again. The process of calculating the release indicator is performed for every keyword and is roughly divided into three parts (see Figure 4.3).

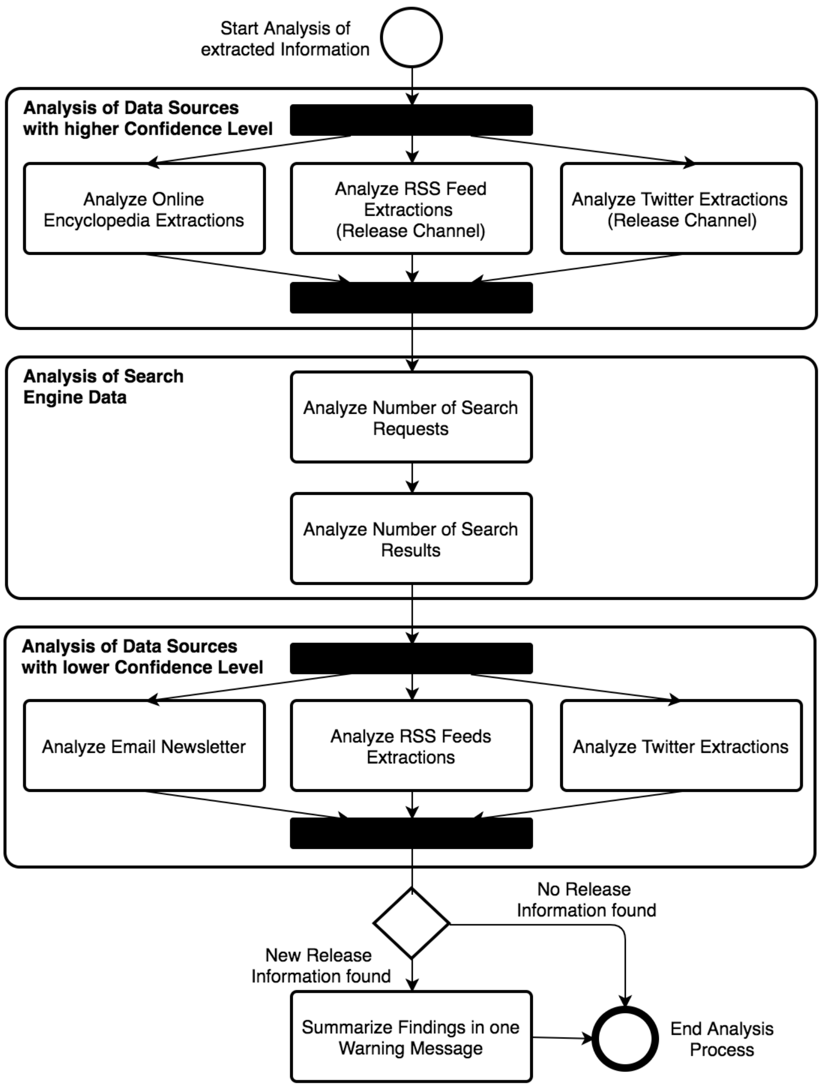

Start Analysis of extracted Information
Analysis of Data Sources with higher Confidence Level
Analyze Online Encyclopedia Extractions
Analyze RSS Feed Extractions (Release Channel)
Analyze Twitter Extractions (Release Channel)
Analysis of Search Engine Data
Analyze Number of Search Requests
Analyze Number of Search Results
Analysis of Data Sources with lower Confidence Level
Analyze Email Newsletter
Analyze RSS Feeds Extractions
Analyze Twitter Extractions
No Release Information found
New Release Information found
Summarize Findings in one Warning Message
End Analysis Process

Figure 4.3: Decision Model for Technology Update Notification

4.3.1 Analysis of Data Sources with high Confidence Level

First, the system analyzes information extracted from data sources with a high confidence level. In our system, there are three types of data sources, which receives a confidence level of 100%: (1) Release information extracted from Wikipedia, (2) RSS feeds, which only exists for publishing new release notes, and (3) Twitter accounts, which also just publish tweets about new technology releases. Whenever one of the mentioned data sources publishes new information, the domain expert is informed directly. Since the system just monitors for changes in this analysis step, the analysis is not based on a specific data processing approach but was developed by ourselves.

4.3.2 Analysis of Search Engine Data

In a second step, data extracted from the search engine Google are analyzed. The extracted data are time series, which is why the system analyzes the data by several statistical approaches. On the one hand, the goal is to find out, if there is a correlation between the search engine user's behaviour and a particular technology release (i.e. number of search requests). On the other hand, an increasing amount of reports about a technology might indicates the announcement or release of a new update (i.e. number of search results).

The analysis of the number of search results is quite simple, since there is just one value per extraction from Google Search. Our assumption is that a distinct increase in the number of search results in compare to the previously extracted values may indicate a technology release.

The analysis of the number of search requests, which are gained from Google Trends as a CSV file, is more complex, since we gain 90 data points per data extraction and the data points are normalized. Due to the complexity of this data analysis, we discuss the process in more detail:

Google provides trend data from the behaviour of their search engine users via Google Trends. The extracted data are time series, for which a custom start and end date can be configured. The obtained measuring points are normalized with 100. Thus, the day with the highest number of search request is represented by 100 and the day with the lowest number of search requests is represented by 0.

To achieve a human like detection of trends in the Google Trends data, we decided to use several statistical parameters in the analysis process of the number of search requests, which are listed below:

- **Moving Average**

  The first analysis step is based on the concept of the moving average. The moving average represent the average of the past n data points. It is calculated by adding all values from the selected data pool and then dividing the sum by the number of values. This process is then repeated with a different data pool and the resulting averages are compared against each other.

Since Google Trends offers normalized data, we adjusted the process of the moving average as follows: we first calculate the average of all data points per data extraction and second, try to detect a trend while comparing the calculated averages.

For example, if we have extracted the CSV file from Google Trends (which contains the amount of search requests for the past 90 days normalized by 100) for a specific keyword for the past 30 days, we end up with 30 averages and each average represents 90 values. We then analyze the trend of the averages over the extracted time period. If the average decreases from one day to the next day over 15 percent, a noticeable increase of search requests are detected. This is the case, since an increase of search requests relativizes values from the days before.

- **Standard Deviation**

  The standard deviation is analyzed with the same procedure but aims at a different goal. Since the Google Trend data for keywords from the technology area often have a high standard deviation, the occurrence of a higher demand for a technology leads into a new maximum, relativizes the values from the days before and shrinks the standard deviation.

- **Total Number of Jumps**

  In addition to the moving average and the standard deviation, we further analyze the number of jumps that are present for each data extraction. Again, if the prototype detects a significant decrease in the number of jumps from previously extracted time series, an update warning is created. This method was developed by ourselves, since the moving average and the standard deviation were to sensible on time series from uncommon technologies. We discuss this problem further under 6.4.3 and 6.4.5.

4.3.3 Analysis of Data Sources with lower Confidence Level

This step in the analysis process is dedicated to analyze information gathered from data sources with lower confidence level. These data sources are either emails, RSS feeds or twitter accounts. The extracted information is text written in natural language. Therefore, the analysis is done by NLP, which is further described in Section 5.4. The goal is to find sentences that actually report about a technology release.

CHAPTER 5

Prototype Architecture and Design

In this chapter, we document various aspects of the implementation work of this thesis, which have been done in order to implement the three generic models introduced in Chapter 4.

Therefore, we first describe the system's architecture and technology stack (see Section 5.1), which both building the basis of the development process. We then describe the structure of the database by explaining the purpose of the entities and their relationships using an entity relationship model (see Section 5.2), which represents both the structure of the database and the model classes.

Second, we document the implementation of the various data extraction methods for the different data sources defined in the Online Source Data Model (see Section 4.2) in Section 5.3.

Third, due to the complexity of analyzing the natural language data extracted from email newsletters, RSS feeds and Twitter posts, we describe the implementation of the NLP pipeline as well as the main challenges occurred during the development process of the NLP tasks in Section 5.4.

5.1 System Architecture

In this section, we describe both the system architecture and the technology stack of the developed prototype.

Before we started implementing the prototype, we had to choose the programming language for the backend. From the three developed models, we derived a list of tasks which had to be developed and integrated in the prototype. After a comprehensive market research, we found supporting Java libraries for all tasks, which is why we have selected Java as a base for the prototype.

In order to achieve a quick implementation of the basic functionalities, we used jHipster[1] to set up the Web application. jHipster allows developers to generate, develop and deploy a web application. The database structure can be configured in the JDL Studio[2], which outputs a model in form of a JDL file. This model is not only used to generate the database structure but creates the code for CRUD[3] functionalities for both the frontend and backend. When creating a web application with jHipster, you will be guided through a step-by-step tutorial, which allows for project specific adjustments. After applying the JDL file to the web application, the basic structure of the web application is set up.

The system architecture together with parts of the technology stack is sketched in Figure 5.1 and can roughly be divided into three areas: frontend, backend and the database server.

The implementation of the frontend was done in Angular 4 and was designed with bootstrap. Yarn[4] was used for frontend dependency management.

The backend is developed in Java 8 and is a Spring Boot[5] application. The Maven[6] configuration handles the build, test and run of the application. The communication to the backend is handled by the controller classes, which are able to receive and answer to Representational State Transfer (REST) calls. The communication to the database is handled by Spring Data JPA[7] repositories. The service layer is not only responsible for forwarding requests from the controllers to the persistence layer and vice versa, but extracts data from various heterogeneous data sources. The extraction methods for each data source are different and are described under Section 5.3 in detail. The backend is secured by Spring security[8], which supports both authentication and authorization, protects against attacks and is integrated with Spring Web MVC. Spring Web MVC is used in this project to provide RESTful web services.

We choose PostgreSQL[9] as a database server because it can be used commercially for free and still provides all the functionalities required for this project. The database runs in a docker container, which allows independency between the infrastructure and the application. In Section 5.2, we describe the entities of the database in more detail.

[1]http://www.jhipster.tech/

[2]https://start.jhipster.tech/jdl-studio/

[3]CRUD is a acronyms for Create Read Update Delete and described the basic interactions with the persistence layer.

[4]https://yarnpkg.com/

[5]https://projects.spring.io/spring-boot/

[6]https://maven.apache.org/

[7]http://projects.spring.io/spring-data-jpa/

[8]http://projects.spring.io/spring-security/

[9]https://www.postgresql.org/

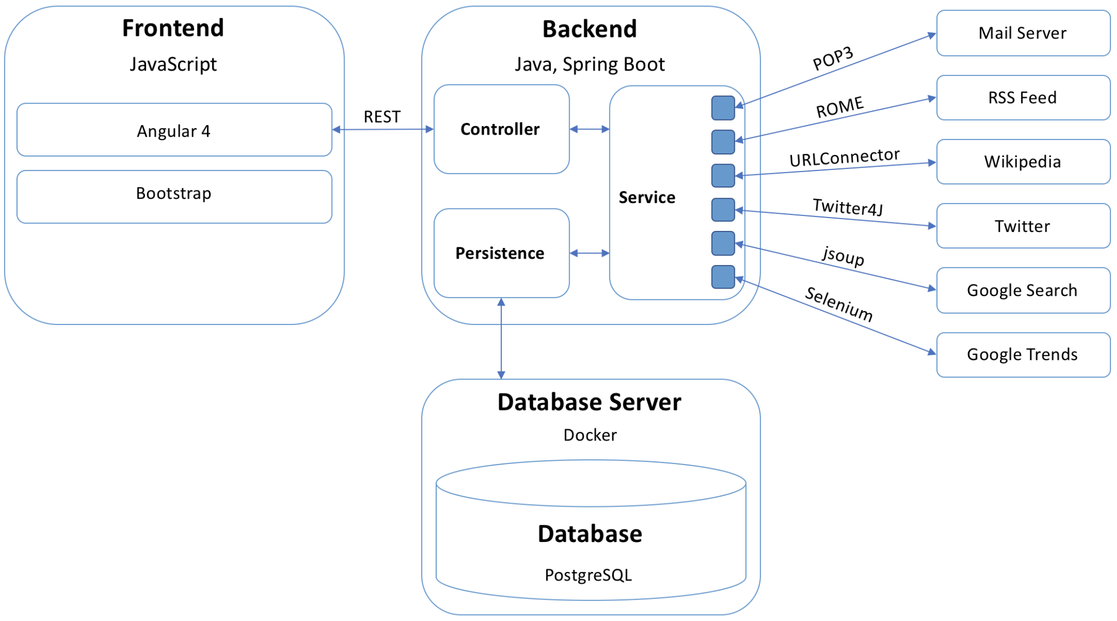


Figure 5.1: System Architecture

5.2 Database

The entity relationship model shown in Figure 5.2 represents both the structure of the database and the model classes. At the beginning of each software prediction process, the domain expert has to define a technology he or she would like to observe. The selected technology is then saved in the entity *Technology*.

After that, for each technology the domain expert has to define *Keywords* which might represent the name or version number of the next release. The keywords should be defined carefully, since the their quality can have an impact on the results of the prediction process.

Since the prototype is capable of sending emails in case of discovering a technology release, the user can create a list of *Contacts* and link them to certain technologies.

For the setup process, the user further has to define the different types of data sources:

- **Email**

  The prototype receives emails from a predefined email address once a day and saves them in the entity *Email*. If an email has already been received from the same sender, the sender is already stored in the entity *EmailSender* and the email will be linked to the entry. If no email was previously received by the sender, a new entry will be created in the entity *EmailSender*.

- **RSS Feed**

  In the entity *RSSFeed*, the name and the URL of the RSS feeds to be observed are configured. On a daily basis, the prototype checks the RSS feeds for new entries and saves them in the entity *RSSFeedEntry.* The domain expert is also capable of marking a RSS feed as release channel, if it just announces new technology releases for a specific technology.

- **Twitter**

  A predefined Twitter account is used to subscribe to other Twitter accounts which might post about a new technology release and is saved in the entity *TwitterAccount.* The related Twitter accounts are visited once a day to extract the new Tweets. New tweets are saved in the entity *Tweet* and are linked to the according *TwitterAccount* entity. Twitter accounts can be tagged as release channel as well (see description of entity RSS Feed).

- **Wikipedia**

  The domain expert further defines Wikipedia pages, which offer release information, including "*Latest release*" and "Preview release", and are linked to a *Technology.* This information is saved in the entity *WikiData.* Every time the release information is updated, the new release version is saved in the entity *WikiDataExtraction.*

The keywords further refer to the *NumberOfSearchResults*, which is extracted directly from Google Search and the *NumberOfSearchRequests*, which is gained from Google Trends. Both the number of search results and the number of search requests are extracted on a daily basis.

If the analysis process of the prototype detects relevant information, a warning is created and saved in the entity *UpdateWarning.* The update warnings are further analyzed by a process, which consider the reason why the warning was created. If the update prediction process of the Decision Model for Technology Update Detection discovers a new technology release, all related update warnings are bundled in the entity *WarningMessage* and emails will be sent to contacts which relate to the technology.

The entity *ConfidenceLevel* saves the level of trustworthiness of an online source (types are saved in *EntityType.* Each data source receives an initial confidence level, which may get adjusted by the feedback of the domain expert.

The enum *ProcessStatus* is used to show the status of an extracted *RSSFeedsEntry*, *Tweet* or *Email.* The status may change over time and depends on whether the data source contains information relevant to a new technology release.

The enum *UpdateWarningStatus* indicates the status of the entity *UpdateWarning.*

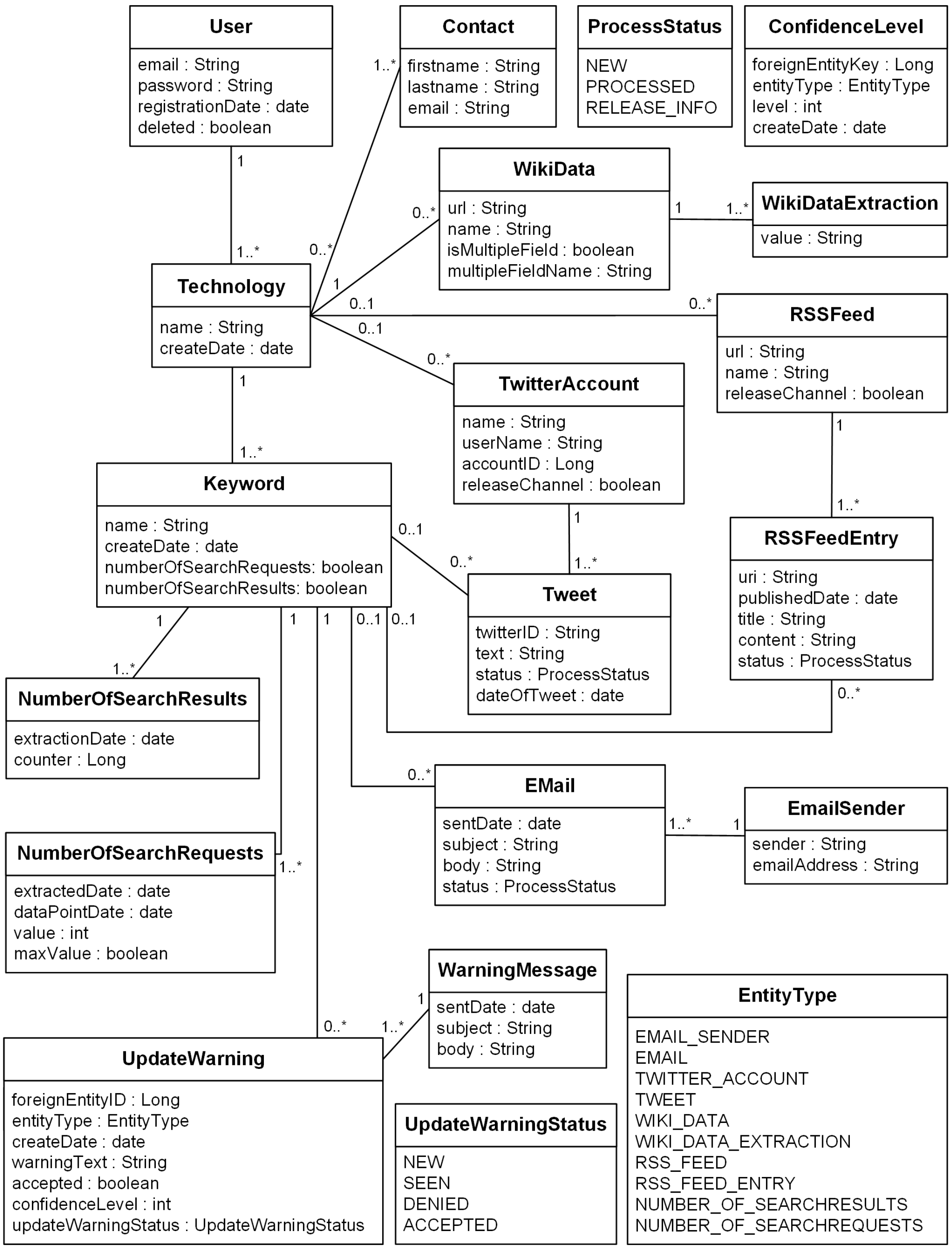


Figure 5.2: Database Model

5.3 Data Extraction Methods

In the course of the development of the prototype different data extraction methods were implemented to gather information. In this section, we present these methods by listing code snippets showing the use of the data extraction technologies.

5.3.1 Email

The emails are received from the email server via POP3[10]. The code of the Java implementation is shown in Listing 5.1. First, the properties for the email server have to be defined. Second, a *Session* with these properties and a *Store* object for connecting to the server is created. Third, a *Folder* object from the *Store* is created to open the inbox in read-only mode. Finally, new emails are received and stored in the database.

```
Properties properties = new Properties();
properties.put("mail.pop3.host", emailHostValue);
properties.put("mail.pop3.port", "995");
properties.put("mail.pop3.ssl.enable", "true");
// Set property values for session
Session session = Session.getDefaultInstance(properties);
// Create POP3 store object and connect with the server
Store store = session.getStore("pop3s");
store.connect(emailHostValue, emailAddress, emailPassword);
// Create folder object and open it in read-only mode
Folder folder = store.getFolder("INBOX");
folder.open(Folder.READ_ONLY);
for(Message message : folder.getMessages()){
    // Check whether Email content is plain text or html
    String messageContent = processMessageContent(message);
    Email email = new ...
    emailService.save(email);
}
folder.close(false);
store.close();
```

Listing 5.1: Receiving Emails from a Mail Server via POP3

Before the email content can be saved in the database, it has to be treated separately (see Listing 5.1, call of method *processMessageContent(...)*), because emails can contain parts with different content types. The Listing 5.2 shows that the content of a message from type *text/plain* or *text/html* can be extracted without further processing. If the message content is from type *multipart*, the content must be cast to MultiPart and then each part can be extracted separately.

[10]Post Office Protocol version 3 (POP3) is an internet protocol to retrieve E-mails from a server.

```java
private String processEmailContent(Message message){
    String messageContent = "";
    String type = message.getContentType();
    if (type.contains("multipart")) {
        Multipart multiPart = (Multipart) message.getContent();
        for (int i = 0; i < multiPart.getCount(); i++) {
            MimeBodyPart part = (MimeBodyPart)
               multiPart.getBodyPart(i);
            messageContent =
               Jsoup.parse(part.getContent().toString()).text();
        }
    }else if (type.contains("text/plain") ||
       type.contains("text/html")) {
        Object content = message.getContent();
        messageContent =
           Jsoup.parse(content.toString()).text();
    }
    return messageContent;
}
```

Listing 5.2: Process Email content based on Content Type

5.3.2 RSS Feed

The extraction of RSS feeds is implemented with the help of the Java framework ROME. The ROME framework is able to parse various types of syndication feeds, which is used to process the RSS feeds. First, the URL of the feeds is specified and then the feed's content can be read by using the *SyndFeedInput* class. Before a new entry is created in the database, it is checked whether the entry already exists. This procedure is represented by the Listing below:

```java
rssFeedService.findAll().forEach(rssFeed -> {
    URL feedSource = new URL(rssFeed.getUrl());
    SyndFeedInput input = new SyndFeedInput();
    SyndFeed feed = input.build(new XmlReader(feedSource));
    feed.getEntries().forEach(syndEntry -> {
        if(rssFeedEntryService.findByUri(syndEntry.getUri())
           == null){
            RSSFeedEntry rssFeedEntry = new ...
            rssFeedEntryService.save(rssFeedEntry);
        }
    });
});
```

Listing 5.3: Iterate through RSS Feeds to find new Entries

5.3.3 Twitter

Twitter offers an API[11] to access its functionalities and content. We used the unofficial Java library Twitter4J[12] to handle the communication between Twitter and the prototype, because it offers both extracting the friend list and receiving Tweets. We have created a Twitter account in order to collect data for the evaluation part of this thesis. We subscribed to other Twitter accounts which publish content about technologies. Before the Twitter API can be used, an app must be created at the Twitter developer console[13]. After the app was successfully created, Twitter offers an individual consumer key, a consumer secret, an access token and an access token secret to handle the communication with its API. The following Listing shows the creation of an object of the class *Twitter*, which used the four parameters mentioned before:

```
ConfigurationBuilder cb = new ConfigurationBuilder();
cb.setDebugEnabled(true)
    .setOAuthConsumerKey(consumerKey)
    .setOAuthConsumerSecret(consumerSecret)
    .setOAuthAccessToken(accessToken)
    .setOAuthAccessTokenSecret(accessTokenSecret);
TwitterFactory tf = new TwitterFactory(cb.build());
Twitter twitter = tf.getInstance();
```

Listing 5.4: Create an Object of Class Twitter

In order to receive Tweets from the subscribed Twitter accounts, the prototype claims the friend list. Since the Twitter API only allows a maximum response length of 20 entries, a *PagableResponseList* and a *Courser* are used to load additional results. This is done by the following code snippet:

```
List<User> userCollection = new ArrayList<User>();
PagableResponseList<User> users =
   twitter.getFriendsList(accountID, -1);
do{
    long courser = users.getNextCursor();
    for(User user : users){
        userCollection.add(user);
    }
    users = twitter.getFriendsList(accountID, courser);
}while(users != null && users.size() != 0);
```

Listing 5.5: Get Friend List for a Twitter Account

The resulting friend list is then investigated to find new Tweets. Therefore, the system checks, whether the friend is already saved as an entry in the database. After that,

[11] https://developer.twitter.com/en/docs/api-reference-index
[12] https://github.com/yusuke/twitter4j
[13] https://apps.twitter.com/

the account's *HomeTimeLine* is claimed, which contains the latest 20 Tweets. The *HomeTimeLine* is then iterated and new Tweets are stored in the database, as can be seen in the Listing below:

```
users.forEach(user -> {
    TwitterAccount twitterAccount =
       twitterAccountService.findByAccountID(user.getId());
    if(twitterAccount == null)
        twitterAccount = new ...

    twitter.getUserTimeline(user.getId()).forEach(status -> {
        Tweet tweet = new ...
        tweetService.save(tweet);
    });
});
```

Listing 5.6: Receive all Tweets from an User List

5.3.4 Wikipedia

The extraction of release information from Wikipedia starts by receiving the technology related Wikipedia page via a *BufferedReader*. This *BufferedReader* reads the content by using an *InputStream*, created from an *URLConnection*. The page is then scanned line by line, until the defined release information is found. The information is then trimmed from unnecessary information by the method *extractReleaseInformation(...)* and saved in the variable *releaseInformation*. The described procedure is shown in the listing below:

```
URL url = new URL(wikiData.getUrl());
InputStream is = url.openConnection().getInputStream();
BufferedReader br = new BufferedReader(new
   InputStreamReader(is));

String line = null;
String releaseInformation = "";
while ((line = br.readLine()) != null) {
    if (line.contains(wikiData.getName())){
        releaseInformation = extractReleaseInformation(br);
        break;
    }
}
```

Listing 5.7: Extract Release Information from Wikipedia

When the current release version was found, it is compared with the one previously extracted. If the comparison shows a delta, it will be saved in the database and an *UpdateWarning* will be created.

5.3.5 Number of Search Results

The extraction of the number of search results from Google Search was implemented with the help of the Java library Jsoup. Jsoup enables the download of Web pages by entering the URL and *UserAgent*. After the document is extracted via the Jsoup framework, the value is extracted from the div which contains the number of search results. Since the value contains additional information (e.g. „*About 15.200.000 results (0,35 seconds)*“), the extracted text has to be trimmed to receive the actual number by calling the method *extractNumberFromElement(...)*. This number is then saved in the database. Since Google checks, whether requests are sent from an system automatically[14], after each iteration the prototype waits randomly between 25 and 100 seconds to prevent being tagged as an automated service by Google. The described procedure of extracting the number of search results from Google Search is listed below:

```
List<Keyword> keywords = keywordService.findAll();
keywords.forEach(keyword -> {
    String url = "https://www.google.com/"+
                "search?q=\""+keyword.getName()+"\"";
    Document document = Jsoup
        .connect(url)
        .userAgent(...)
        .get();
    Element element =
       document.select("div#resultStats").first();
    Long number = extractNumberFromElement(element);
    NumberOfSearchResults nsr = new ...
    numberOfSearchResultsService.save(nsr);
    waitRandomly();
});
```

Listing 5.8: Receive the Number of Search Results from Google Search

5.3.6 Number of Search Requests

The information about the number of search requests are received from Google Trends, which functionalities are discussed in Section 2.3.2. Since Google Trends does not provide an API for extracting the normalized number of search requests, the prototype uses Selenium[15] to download the CSV file. The procedure works as follows: before the prototype is able to open the browser, the *WebDriver* must be set. Then, for each keyword selenium opens the browser and visits the URL. The URL contains the search parameters and looks as follows:

`https://trends.google.com/trends/explore?date=today%{}203-m&q=keyword`

[14] https://support.google.com/websearch/answer/86640?hl=en
[15] http://www.seleniumhq.org/

The first GET parameter *date* has the value *today 3-m* which means that the CSV file should contain the numbers of search requests for the last 3 months. The second parameter q contains the keyword.

After the site is loaded, the button for downloading the CSV file is clicked by selenium. The downloaded file is then processed by the method *importCSVData*. This method saves the numbers of search requests per day in the database. Again, the prototype waits randomly to avoid getting marked as automated service by Google. The following listings shows the selenium tasks:

```
System.setProperty("webdriver.chrome.driver", path);
WebDriver driver = new ChromeDriver();
keywords.forEach(keyword -> {
    driver.get(keyword.getURLGoogleTrend());
    // wait for the line chart div
    WebElement element = (new WebDriverWait(driver, 30)).
        until(ExpectedConditions.elementToBeClickable(
        By.cssSelector("div.fe-line-chart-header-title")));
    // select csv download button and click it
    driver.findElement(
        By.cssSelector("button.widget-actions-item.export")).click();
    // wait until the download has finished
    Thread.sleep(15000);
    importCSVData(keyword, ZonedDateTime.now());
    waitRandomly();
});
driver.quit();
```

Listing 5.9: Receive the Number of Search Requests from Google Trends

5.4 Application of NLP

In the previous section (see 5.3), we described, among other things, how the system extracts information from email newsletters, RSS feeds and Twitter posts. These three data sources have in common that they provide information in the form of natural language. In order to find information about an (upcoming) technology release in natural language, NLP is used to perform automated processing and analysis. In this section, we describe the three-stage NLP pipeline, which is designed to detect technology releases.

Before we started implementing the NLP pipeline, we thoroughly analyzed the extracted texts. The goal was to find as many variations as possible of how to communicate (upcoming) technology releases. The NLP pipeline resulting from the formulations is split into three stages. First, the prototype splits texts into sentences (see Task 1 in Section 5.4.1). Second, these sentences are analyzed by regular expression (see Task 2.1 in Section 5.4.2). In Task 2.2, the sentences are analyzed by using certain NLP tools (see

Section 5.4.3). In a third stage, the prototype tries to match the found release messages to certain keywords. If a keyword is detected in one of the release messages, an update warning will be created. Figure 5.3 illustrates the NLP pipeline:

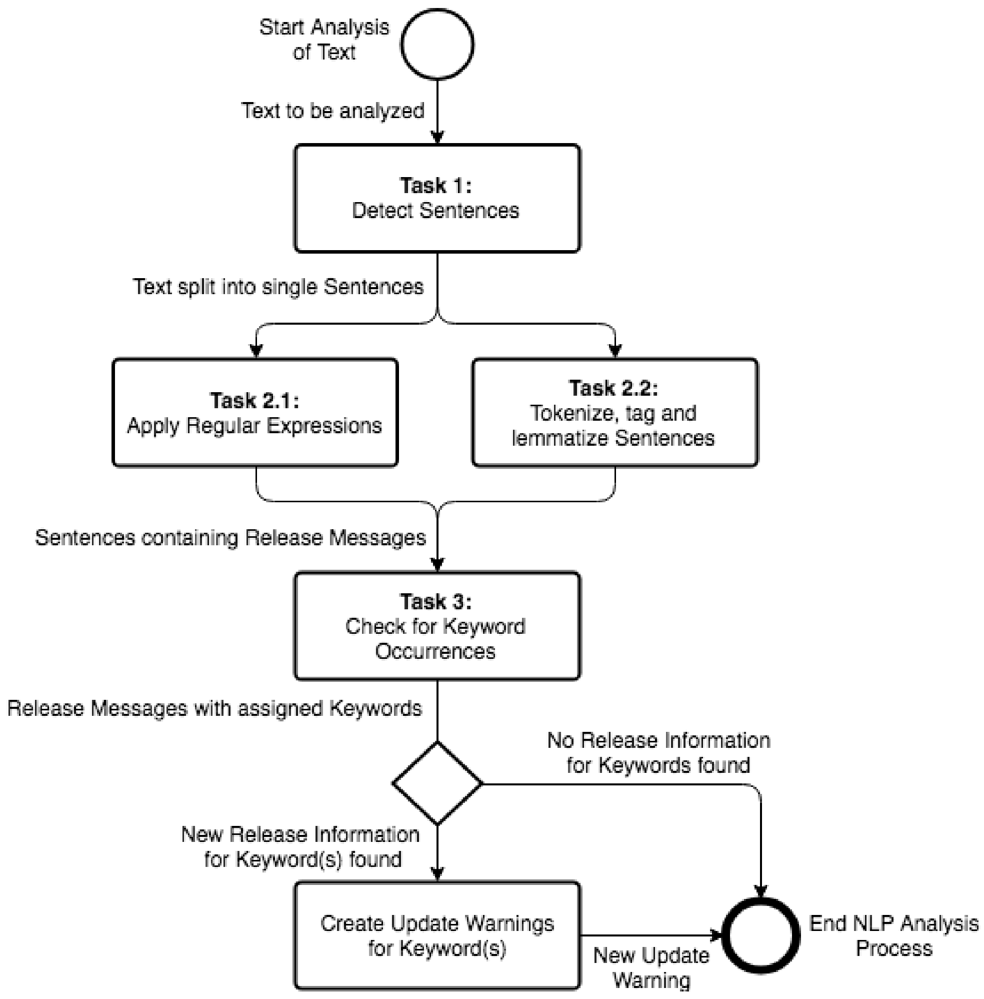

Start Analysis of Text
Text to be analyzed
Task 1: Detect Sentences
Text split into single Sentences
Task 2.1: Apply Regular Expressions
Task 2.2: Tokenize, tag and lemmatize Sentences
Sentences containing Release Messages
Task 3: Check for Keyword Occurrences
Release Messages with assigned Keywords
No Release Information for Keywords found
New Release Information for Keyword(s) found
Create Update Warnings for Keyword(s)
New Update Warning
End NLP Analysis Process

Figure 5.3: NLP Pipeline

Due to the complexity of the tasks 1, 2.1 and 2.2, we describe them further in the following sections.

5.4.1 Task 1: Split Texts into Sentences

The basis for the analysis via NLP are the texts extracted from the data sources email, RSS feeds and Twitter. These data sources often do not provide individual sentences, but longer texts. In order to analyze these texts to find potential release messages, we - in a first step - split them into single sentences. The theoretical part of this work has already shown that a division into sentences sounds trivial, but is not easy to implement (see Section 3.4.1). This task was especially difficult due to the domain and the selected data sources. We have split the sentence delimiting into four subtasks:

1. First, the prototype removes special characters, such as emojis, no-break spaces[16] or zero width spaces[17], since both regular expressions and NLP tools cannot gain further information from them.
2. Second, the prototype splits the texts by identifying URLs.
3. Third, the prototype detects sentences by punctuation characters (i.e. “.“, “?“, “!“).
4. Finally, the prototype tries to detect sentences, which do not end with a punctuation character.

To fulfil all these steps, we had to implement an own sentence delimiter, since the one from Apache OpenNLP was not capable of handling the context specific exceptions. The following list comprises the main challenges we had to face while implementing the sentence delimiter:

- **Detecting URLs**

  We tried using URL detectors from various sources but ended up implementing our own one which suits the context of the thesis. The prototype detects URLs by using the following regular expression:

  ```
  (http[s]?:[/][/])?(www[.])?[a-z,A-Z,0-9,-,\\.]+[.]
  (-- various domains --){1,2}[^ ,;]*
  ```

  Listing 5.10: Regular Expression for detecting URLs

  This regular expression is capable of detecting URLs, which may start with the protocol (i.e. http and https) and/or www, having a valid domain name, end with a top level domain (such as .com, .org, .at, and so forth) and might contain a further path, which does not contain a space, comma or semicolon.

- **Handling Version Numbers**

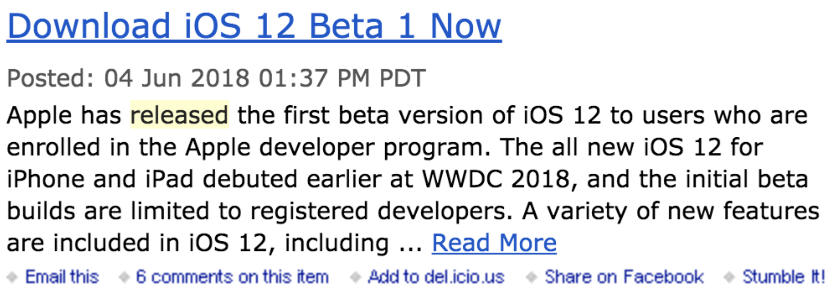
  The sentence delimiter from Apache OpenNLP splits texts by punctuation characters, no matter in which context they appear. This is a problem in the context of this thesis, since one of the goals is to identify sentences which reports an (upcoming) release and they probably contain a version number as well. Version numbers differ from technology to technology, but often contain a full stop (e.g. NodeJS 9.10.1, PostgreSQL 10.4, Perl 5.27.11 or Python 3.7.0b5). Thus, the sentence delimiter checks, in which context the full stop is used to prevent splitting texts in between version numbers.

[16]Unicode no-break space: U+00A0

[17]Unicode for zero width space: U+200B

- **Detecting Sentences without a Punctuation Character**

  During the analysis of the extracted texts we found that sentences often do not end with a punctuation character. To illustrate this problem, Figure 5.4 shows a screenshot from a newsletter we have received during the evaluation process. The email reports on the release of a beta version of iOS 12.

  Download iOS 12 Beta 1 Now

  Posted: 04 Jun 2018 01:37 PM PDT

  Apple has released the first beta version of iOS 12 to users who are enrolled in the Apple developer program. The all new iOS 12 for iPhone and iPad debuted earlier at WWDC 2018, and the initial beta builds are limited to registered developers. A variety of new features are included in iOS 12, including ... Read More

  Figure 5.4: Screenshot: Email Newsletter OS X Daily from 5.6.2018

  The prototype receives this email (see Section 5.3) and saves it in the database. Unfortunately, nowadays emails are often formatted in HTML, which can make the extraction of the pure text a difficult task.

  As can be seen in the screenshot, the headline of the article says "*Download iOS 12 Beta 1 Now*", which indicates the release of a new beta version of Apple's operating system and is therefore very interesting for our investigation. However, since the headline does not end with a punctuation character, it is merged with the text below and we end up with "*Download iOS 12 Beta 1 Now Posted: 04 Jun 2018 ...*". To capture this and similar cases (e.g. "*Read more*", "*View this*", or "*Click here*"), we have developed regular expressions that are capable of handling these issues and split the extracted texts accordingly.

5.4.2 Task 2.1: Find Release Messages by Regular Expressions

After the texts has been delimited into sentences, the prototype searches for release phrases by making use of various regular expressions. Task 2.1 is split into 3 subtasks:

1. First, the prototype applies three regular expressions to find release messages. If a sentences matches to one of them, it is marked as potential release message.
2. In a second step, the prototype excludes potential release messages, if they contain certain keywords. This exclusion process is again done by matching regular expressions containing keywords, such as "weekly newsletter", "podcast", "book" or "sponsor".
3. In a final step, the remaining sentences are examined for technologies. If one of the technologies to be monitored is in one of the release messages, an update warning is generated.

In the following listing, we explain the regular expressions from the first subtask in more detail:

- **Regular Expression 1: Version Number + Release Phrase**

  The first regular expression looks up for sentences, which on the one hand contain a version number (e.g. 4.0.7), and on the other hand contain some kind of release phrase (e.g. “is out“, “is here“, “is available, “has been released“, etc.), which is behind the version number. Table 5.1 comprises a list of release messages extracted from Twitter, which fulfil the restrictions of the first regular expression:

Date	**Username**	**Tweet**
31.07.2017	@rails	*Rails 5.0.5 is out!*
03.08.2017	@php.net	*PHP 7.2.0 Beta 2 released.*
09.01.2018	@PythonInsider	*Python 3.7.0a4 is available for testing*
21.03.2018	@java	*Today #java 10 will be released!*
19.04.2018	@MySQL	*MySQL 8.0 GA is here!*
26.04.2018	@fedora	*Fedora 28 will be released on May 1st, hitting its original scheduled release (...)*
09.05.2018	@springcentral	*Spring Cloud Data Flow 1.5 RC1 released!*
04.06.2018	@Linux_Mint	*Linux Mint 19 Tara MATE – BETA Release*

Table 5.1: Tweets captured by Regular Expression 1

- **Regular Expression 2: Release Phrase + Version Number**

  The second regular expression looks up again for a version number. But, in contrast to the first regular expression, here the release phrase has to be in front of the version number. Table 5.2 shows Tweets which are matching the described pattern:

Date	**Username**	**Tweet**
06.07.2018	@official_php	*Next in today’s set of releases: #PHP 7.1.7.*
05.02.2018	@IBMZ	*Get all the details on the new release of z/OS Connect EE V3.0.5*
27.02.2018	@springcentral	*We are pleased to announce the release of #SpringCloudDataFlow 1.4.0.M1.*
27.04.2018	@angular	*We’ve just released AngularJS 1.7.0-rc.0.*
05.05.2018	@postgresql	*(...) guidance to upcoming releases, such as PostgreSQL 11 Beta 1:*
31.05.2018	@java	New *#Gradle Releases Version 4.7 with #Java10 runtime support*

Table 5.2: Tweets captured by Regular Expression 2

- **Regular Expression 3: Release Phrase**

  In contrast to the regular expressions introduced before, the third one focuses more on the different types of release phrases, rather than finding concrete version information. Table 5.3 comprises a list of Tweets, which are fulfilling the pattern matching the third regular expression.

Date	**Username**	**Tweet**
20.02.2018	@IBMcloud	*For the first time ever, @SQLServer is now available on #Linux operating systems.*
23.02.2018	@springcentral	*New Spring for Apache Kafka and Spring Integration Kafka releases are available.*
28.02.2018	@AndroidDev	*In the next release of Android, we plan to improve user and developer (...)*
21.03.2018	@nodejs	*Happy @nodejs Current Release Day*
29.03.2018	@SQLServer	*We are excited to announce the March release of #SQL Operations Studio is now available.*
11.04.2018	@ubuntu	*With only two weeks to go until Ubuntu 18.04, here are the latest updates (...)*

Table 5.3: Tweets captured by Regular Expression 3

5.4.3 Task 2.2: Find Release Messages by NLP Tools

After the texts have been split into sentences and the sentences have been analyzed by regular expressions, the prototype then uses certain NLP tools to detect further release messages.

Before we started implementing the NLP tasks with certain NLP tools, we had to choose a NLP system that met our needs and was easy to integrate into the prototype. The following systems were shortlisted: GATE[18], The Standford CoreNLP[19], Natural Processing Toolkit (NLTK)[20] and Apache OpenNLP[21].

We decided to implement the NLP tasks with Apache OpenNLP, because this system offers all features we need for our investigation. Furthermore, it can be easily integrated via Maven into a Java backend. The following list comprises the used NLP tools, which functionality is demonstrated by applying them to this example sentence: "*Apache Tomcat 8.5.23 has been released*"[22].

[18]https://gate.ac.uk/

[19]https://stanfordnlp.github.io/CoreNLP/

[20]https://www.nltk.org/

[21]https://opennlp.apache.org/

[22]Tweet from Twitter user @TheApacheTomcat on 03.10.2017

- **Tokenizer**

  The tokenizer was used to separate the sentences into single words. For our purpose, the *SimpleTokenizer*, offered by Apache OpenNLP, has been applied. It separates character streams by whitespaces. Furthermore, punctuation characters are separated from the last word of a sentence, which is a bonus.

  The first column of Table 5.4 shows the tokens, after applying the *SimpleTokenizer* to the example sentence. For more information about how tokenizers work, see Section 3.4.1.

- **POS Tagger**

  After tokenizing the sentences into words, the POS tagger then assigns suitable tags[23] according to the type of the word (e.g. noun, verbs, or articles) to the words. The second column of Table 5.4 shows the tags assigned to the tokens of the example sentence. For more information about how POS taggers work, see Section 3.4.2.

- **Lemmatizer**

  In a last step, the lemmatizer analysis the tokens together with its assigned POS tags to find the root form of the words (e.g. “release“ is the root form of “released“). The third column of Table 5.4 shows the results of the lemmatizer. If the lemmatizer is not able to find a suiting root form of a word, it returns “0“. For more information about lemmatizing sentences, see Section 3.4.2.

Token	**Tag**	**Lemmatize**
apache	RB (Adverb)	0
tomcat	RB (Adverb)	0
8	CD (Cardinal number)	0
.	.	0
5	CD (Cardinal number)	0
.	.	0
23	CD (Cardinal number)	0
has	VBZ (Verb, 3rd person singular present)	have
been	VBN (Verb, past participle)	be
released	VBN(Verb, past participle)	release

Table 5.4: Apache OpenNLP: Tokenizer, Part-of-Speech Tagger and Lemmatizer

The above listing describes the application of the tokenizer, POS tagger and lemmatizer. The implementation in Java by using Apache OpenNLP looks as follows:

[23] A list of tags can be found under `http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html` (visited on 16.06.2018)

```java
// initialize Tokenizers
SimpleTokenizer st = SimpleTokenizer.INSTANCE;
// initialize Part-of-Speech Tagger
POSModel pm = new POSModel(is);
POSTaggerME pt = new POSTaggerME(pm);
// initialize Lemmatizer
DictionaryLemmatizer dl = new DictionaryLemmatizer(is);

// 1. step: Tokenize sentence
String [] tokens = st.tokenize(sentence);
// 2. step: Tagging tokens
String [] posTags = pt.tag(tokens);
// 3. step: Lemmatize tokens together with tags
String [] lemmatizedTokens = dl.lemmatize(tokens, posTags);
```

Listing 5.11: Applying NLP Tools from Apache OpenNLP

Before we apply the several NLP tools to a sentence, we had to initialize the tokenizer, POS tagger and lemmatizer. We can then apply the tools to the sentence, which we already described in the bullet list above.

CHAPTER 6

Data Source Discussion

In the previous chapters, we introduced the design and the implementation of the prototype in detail. Particularly important here were the three models, namely (1) the Process Model for Technology Update Detection (see Section 4.1), (2) the Categorization Model for Online Data Sources (see Section 4.2), and (3) the Decision Model for Technology Update Notification (see Section 4.3), which build the prototype's basis. To answer the question of whether automated detection of technology releases (in advance) is possible by harvesting online sources, in this chapter we will discuss the evaluation of the data sources.

In order to evaluate the prototype sufficiently, we first discuss the findings related to the data sources from the Categorization Model. More specific, the evaluation of the data source *Technology News* and *Social Media* are discuss together under one Section 6.1, since both are analyzed by NLP. The data source *Online Encyclopedia* is investigated in Section 6.2. The last data source, *Search Engine Data*, is split into two sections. Under Section 6.3, we discuss the extraction of the number of search results from Google Search. Section 6.4 shows the various statistical functions which were implemented to analyze the number of search requests extracted from Google Trends.

6.1 Data Source: Technology News and Social Media (NLP)

In this section, we are going to evaluate the data source *Technology News*, which is represented by email newsletters and RSS feeds, and *Social Media*, which is represented by Twitter. These two data sources are combined under one evaluation process, since both are offering data in form of natural language and therefore were analyzed by NLP.

Therefore, we first discuss the NLP pipeline (see Section 6.1.1) in two ways: (1) Qualitative by showing the weaknesses of both analyzing sentences via regular expressions and NLP tools. In more detail, we point out the limitations of both approaches by evaluating the false positive and false negative diagnoses. (2) Quantitative by comparing the number of found release messages and the accuracy of both approaches. In a second step, we investigate the three data sources (email newsletters, RSS feeds and Twitter) to point out their advantages and disadvantages (see Section 6.1.2).

6.1.1 Compare Regular Expression against NLP Tools

In Section 5.4, we described the two types of NLP processing developed (regular expressions and NLP tools) together with examples for actual release messages found by each approach. In this section, we compare the NLP analysis via regular expressions against the analysis via NLP tools to show, which approach leads into a more accurate result set of release relevant messages.

Regular Expression

The analysis process via regular expressions focuses not on finding individual release related words, but on concrete release phrases. Before the regular expressions can be applied, the prototype splits the texts in single sentences. After that, three different regular expressions for three types of communicating an (upcoming) release are used to identify release messages.

The first regular expression searches for release messages, which contain a concrete version number (e.g. "*5.0.1*") followed by a release phrase (e.g. "*has been released*", "*is here*", etc.). The second regular expression searches for a version number and a release phrase too. However, in this case the release phrase has to be in front of the version number (e.g. "*announcing the release of ...*"). The third regular expression investigates for release messages, which do not contain a version number but some kind of release phrase.

In Section 5.4.2, we already mentioned some examples of Tweets which were found by applying the regular expressions and are actual release messages. For that reason, we now focus more on the weaknesses of applying regular expressions by analyzing the false positive and false negative analyses.

Although the prototype is looking for concrete release phrases, there may be false positive diagnoses due to the fact that formulations about releases can be used in another context as well. Table 6.1 comprises examples of sentences which the prototype has assigned a false positive diagnose while using regular expressions.

The first example contains the release phrase "*is now available*", which was used to communicate the release of a security guide. The second example uses the release phrase "*to announce that our next*" to report about a meetup and the Tweet in the third row reported about a nearly published episode by using the release phrase "*is out*". The forth Tweet reported about a release of the operating system Debian by using the release phrase "*was released*". Unfortunately, it did not report about an update released a few days ago but in 1996, which is why it should not be marked as a release message relevant to the current time.

Date	**Username**	**Tweet**
21.02.2018	@SUSE	*A comprehensive #security guide for #SAPHANA is now available (...)*
12.06.2018	@angular	*We are excited to announce that our next meetup will be on June 27th*
13.02.2018	@RedHatNews	*Episode 4 of Command Line Heroes is out (...)*
28.03.2018	@debian	*Debian 1.2 (Codename Rex) was released in 1996.*

Table 6.1: Tweets found by Regular Expressions with False Positive Diagnose

In contrast to the Tweets from Table 6.1, a similar formulation can also be used for actual release messages. Table 6.2 comprises a list of Tweets which used the same or similar release phrases as the Tweets from Table 6.1, but are actual release messages:

Date	**Username**	**Tweet**
04.04.2018	@RedHatNews	*Fedora 28 Beta is now available.*
12.03.2018	@springcentral	*We are pleased to announce the release of #Spring-CloudDataFlow 1.4.0.RC1.*
20.12.2017	@gradle	*Gradle 4.4.1 is out.*
08.06.2018	@fedora	*The 4.17 Linux kernel was released earlier this week.*

Table 6.2: Tweets found by Regular Expressions with True Positive Diagnose

The first Tweet of Table 6.2 reported about the release of a new beta of the operating system Fedora by using the release phrase "*is now available*". The second example shows the use of the release phrase "*to announce the release*" for reporting about a new version of a Spring product. The third Tweet stated that a new version of Gradle has been released, which was done by using the release phrase "*is out*". And the last example communicated an update to the Linux kernel by using the release phrase "*was released*".

In addition to the false positive diagnoses, regular expressions may also apply a false negative diagnose, due to special cases or unusual formulations. Table 6.3 comprises a list of Tweets, which were not marked as release messages by the use of regular expressions, but are actual containing information about a release:

The first example was not detected as release message, since the word release is not separated from the version number and the second part of the sentence uses the phrase "*we are giving*", which is quite unusual and not part of one of the regular expressions. The second example reported about an upcoming release of Apache Lucene, but the keyword for the release phrase was packed into other words (i.e."*Multi-Release-JAR*") and therefore not detected by the regular expressions. And the third false negative example was not detected by the regular expressions, since the Tweet just containes the word "*release*", but not a release phrase.

Date	Username	Tweet
02.03.2018	@springcentral	*On the heels of Spring Boot 2.0.0.RELEASE we are giving you @SpringCloud Stream 2.0.0.RC2.*
09.02.2018	@java	*Apache #Lucene 7.3 will use the Multi-Release-JAR feature (...)*
21.05.2018	@plantepostgres	*Andrew Dunstan: PostgreSQL Buildfarm Client Release 8 (...)*

Table 6.3: Tweets found by Regular Expressions with False Negative Diagnose

From the Tweets of the Tables 6.1 and 6.2 we can conclude that the same or similar release phrases can be used for both reporting about an actual release and communicating about a subject not related to a technology release, which may lead into false positive diagnoses. Furthermore, from Table 6.3 we can conclude that information about releases may be communicated in various formulations. Covering most of the different types of these formulations is associated with constant extensions. However, each extension has the potential of increasing the number of false positive diagnoses, which should be considered too.

NLP Tools

The investigation via NLP tools focuses more on concrete occurrences of release related words, rather than release phrases. After the extracted texts are split into single sentences, the prototype prepares the search for release related words by tokenizing, tagging and lemmatizing the sentences. The output of these tasks is a sentence split into words, and for each word, the root form together with its tag (e.g. adverb (RB), verb in past participle (VBN), etc.) was computed (see Section 5.4.3 for an example). Since the prototype should detect information about an (upcoming) technology release, it then searches for specific verbs, such as "*update*", "*release*", "*publish*", "*announce*" and "*available*", as well as nouns, such as "*update*", "*release*" and "*version*" in their root form.

Similar to the analysis via regular expressions, analyzing sentences which may contain release messages can lead to the same problems: false positive and false negative diagnoses.

In contrast to the analysis per regular expressions, the NLP tools just analysis one word and its root form. This leads into a situation where the context, in which the release word is used, may differ from a release message. Table 6.4 comprises a list of Tweets, which are detected by the application of NLP tools, but do not report about a release:

The first example was detected by the prototype, since it contains the word "*announced*". However, the tweet did not report about an upcoming SAP release but reported about financial results. The second example contains the release keyword "*updated*". This keyword was not used in the context of announcing a new release, but communicating an edit of a collection of applications for Ubuntu. The last example was detected by the prototype, since the sentence contains the word "*update*". Unfortunately, the keyword was not related to the technology, but to a video which reports about localization and internationalization for two projects.

Date	**Username**	**Tweet**
30.01.2018	@sapnews	*SAP today announced preliminary financial results for the fourth quarter and full-year 2017:*
02.03.2018	@ubuntu	*It's Friday, so we've updated the Featured Applications & Editor's Picks in #Ubuntu Software*
16.04.2018	@nodejs	*3-minute update on the internationalization and localization of the @electronjs and @nodejs projects.*

Table 6.4: Tweets found by applying NLP Tools with False Positive Diagnosis

In addition to that, using NLP tools for finding release messages by just one word can lead into false negative diagnoses too. Table 6.5 shows some examples, which were not detected by the NLP tools, but actually containing release relevant information:

Date	**Username**	**Tweet**
14.02.2018	@springcentral	*Spring Cloud Task 2.0.0.M3 is out!*
11.03.2018	@RedHatNews	*#RedHat Satellite 6.3 & #RedHat #CloudForms 4.6 are here.*
08.06.2018	@MySQL	*My slides for my talk: "OMG MySQL 8.0 is out!"*

Table 6.5: Tweets not captured by applying NLP Tools

The first example of Table 6.5 reported about an update for Spring Cloud Task, which was communicated by the release phrase "*is out*". Since the NLP tools just search for single words related to the release context, this type of release phrase cannot be detected by the developed NLP pipeline. Both the second Tweet, which communicated an update of two systems from RedHat and the third Tweet, which indirectly reported about a new version of MySQL, haven't been detected for the same reasons mentioned before.

The approach of analyzing single words of sentences via NLP Tools to find release messages faces the same problem as the approach of regular expressions: words which communicate a release are not just reserved for this context, which can lead to false positive diagnoses as mentioned in Table 6.4. Furthermore, since the developed NLP pipeline just analyzes single words, releases phrases, such as "*is out*" or "*are here*", cannot be detected.

From analyzing the Tweets found by the NLP tools we conclude that this approach can detect release messages. But, on the one hand, release related keywords are used in another contexts, which can result in a false positive diagnose. On the other hand, false negative diagnoses are possible as well, since the NLP tools just analyze single words.

Evaluation of the extracted Texts

After comparing the two approaches in a qualitative way by analyzing their limitations and weaknesses, we continue the comparison in a quantitative way by analyzing their found release messages.

In order to compare the two approaches, we collected 22002 Tweets from 61 Twitter accounts between January and June 2018. The extracted 22002 Tweets were split into 42.490 individual sentences by the developed sentence delimiter. From these 42.490 sentences, 746 contain information about an (upcoming) release.

Table 6.6 summarizes the application of the two approaches. In the first row, we compare the number of messages which were marked as release messages by the respective approaches. The analysis via regular expressions has marked 988 sentences as release message, the analysis via NLP tools has marked 1709.

From the 988 sentences detected by the first approach, 722 are actually reporting about a release, which leads to a false positive rate of 27 % (266 false positive diagnosis from 988 sentences marked as release message) and a false negative rate of 3 % (24 of 746 actual release messages were not detected).

From the 1709 sentences detected by the second approach, 704 are actually release messages, which leads to a false positive rate of 59% (1005 false positive diagnosis from 1709 sentences marked as release message) and a false negative rate of 6% (44 of 746 actual release messages were not detected).

	Approach 1: Regular Expression	**Approach 2: NLP Tools**
#Release Messages found	988	1709
#Actual Release Messages	722	704
False Positive	266	1005
False Negative	24	44

Table 6.6: Compare Regular Expressions and NLP Tools

In summary, it can be said that by the application of NLP, release messages can be detected. However, a 100% accurate result cannot be achieved, since both concrete release phrases and just single release related words can be used in other contexts as well. Our evaluation has shown that with the help of regular expressions, the prototype was capable of finding 96.7% of the actual release messages, while the approach of NLP tools lead to a success rate of 94.3%. A much larger difference has been found in the evaluation of the false positive rate: the analysis via regular expressions has a false positive rate of 27%, whereas the analysis via NLP tools of 59%.

6.1.2 Appropriateness of the NLP Data Sources

In this section, we compare the extracted information from those data sources of the Categorization Model (see Section 4.2) which offer data in natural language. For each data source, we analyze the simplicity of extracting the information, as well as its timeliness and quality. During the evaluation phase of this thesis, the prototype extracted data in form of natural language from the following three data sources:

- **Email**

  Receiving data via email for further text processing was easy to implement (see Section 5.3.1). Listing 5.1 shows, how emails can be received via POP3 by a Java application. However, processing emails which do not consists of plain text, but rather complex HTML and JavaScript content, is more difficult and described in Listing 5.2. In contrast to social media platforms, emails come with the advantage that the number of accesses and data extractions from the inbox is not limited.

  Another advantage of email newsletters is that they are still widespread and sometimes specialized in one technology. This can lead to very detailed reports of news in the area of a single technology. However, this advantage comes with a disadvantage: since there is no limitation in terms of text length, the key messages of an article can by distributed over various sentences or even worst paragraphs, which makes the automatic text processing and especially understanding the relations between sentences a difficult task.

  With the nearly unlimited length of texts in an email newsletter comes another disadvantage: as described in Section 5.4.1, emails can be formatted in a way that they contain a list of articles, where each element consists of a headline, a short description of the article, a link to the website and some meta information (e.g. see Figure 5.4). This can cause a problem, since headlines and other text blocks may not end with a punctuation character. However, the implementation of a good sentence delimiter provides the basis for efficient and accurate text processing, which is why the described circumstance makes the implementation of a sentence delimiter for splitting texts into single sentences a tough task.

- **RSS Feeds**

  Just like email newsletters, extracting RSS feed entries is easy to implement. In Section 5.3.2 we have shown, how the prototype extracts entries from RSS feeds by the help of the Java framework ROME. And as before, there are no limitations in regard of the accessibility and the amount of extractions from a RSS feed.

  Since RSS feeds are offered in XML, they are better structured than emails and therefore do not have the problem of implementing an accurate sentence delimiter. Furthermore, there are RSS feeds which are just used for communicating new releases, such as `https://www.postgresql.org/versions.rss`. This can make the process of detecting a new release an easy task. However, they are also not limited in terms of text length, which can lead to a distributed key message over multiple sentences and thus, the understanding of the key message via automatic text processing a complicated task.

RSS feeds come with another disadvantages: for an accurate evaluation of the prototype, we searched for RSS feeds for 68 products from 40 manufactures. But we were not able to find more than 18 RSS feeds which reported about the products of these 40 manufactures. It seems that manufactures focus more on email and newer communication media, such as social media platforms, rather than RSS feeds. Furthermore, the found RSS feeds mostly report about established manufacturers, such as Oracle, Microsoft or RedHat (for a full list, see Attachment A.2), and not about newer technologies.

- **Twitter**

  Twitter was chosen to represent the category social media. Before we could start with the implementation of the data extraction, we had to create a Twitter account and unlock it for data extraction via an API by creating an app at the Twitter developer console. The implementation of the data extraction method was quite easy thanks to the API Twitter4J (see Section 5.3.3).

  Since tweets are limited to just 280 characters, Twitter has two distinct advantages over email newsletters and RSS feeds: first, limiting Tweets to 280 characters forces the user to write short texts and to focus on the main message. Thus, user usally write in short messages and do not produce much overhead. Second, since the key message cannot be distributed over multiple sentences, the implementation of the automatic text processing is easier.

  Another advantage of social media platform is the community, which allows interaction between people. This interaction can lead to more up-to-date information and enables a discussion about (upcoming) releases, which may help in detecting technology updates earlier. The main disadvantage of using Twitter as a data source is its limitation to requests for standard users.

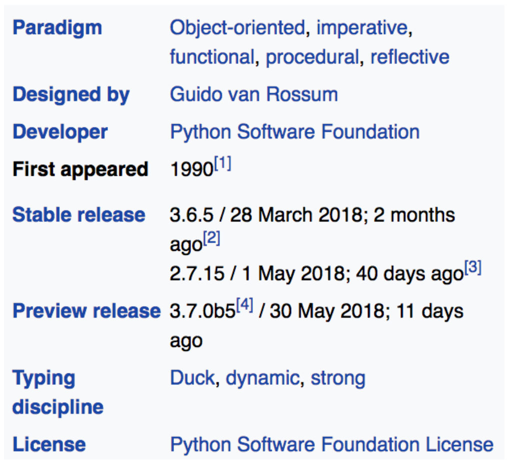
In summary, it can be said that each data source comes with advantages as well as disadvantages. However, our evaluation has shown that due to its compromised content on 280 characters and the huge community, Twitter is the most suitable data source of the developed Categorization Model in order to gain natural language to detect release messages.

6.2 Data Source: Online Encyclopedia

The development of the prototype aimed to create a system that monitors predefined technologies to detect the release of new updates as early as possible. However, not every technology generates the same amount of content or is not equally relevant to the public and domain experts. This can lead to the situation that the analysis of extracted Web content via NLP (see Section 6.1) and the analysis of the number of search requests (see Section 6.4) do not find information about an upcoming technology release, especially if we look for minor updates.

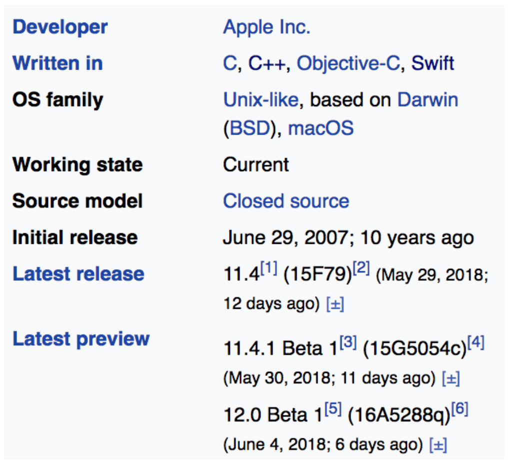
In order to cover these cases as well, the extraction of release information from Wikipedia, which represents the category online encyclopedia, has been integrated into the Categorization Model of the prototype. Wikipedia often offers an info box in their articles, where the users can find a selection of properties of the suspect. Figure 6.1 shows the info box of the Wikipedia page from Python[1], while Figure 6.2 shows the info box of the Wikipedia page from iOS[2].

The comparison of these two info boxes shows that the information within the info boxes can differ significantly and depends on the subject. Even the information about a release is sometimes named differently. For example, the two screenshots show five different release information: stable release, preview release, initial release, latest release and latest preview.



Paradigm	Object-oriented, imperative, functional, procedural, reflective
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	1990[1]
Stable release	3.6.5 / 28 March 2018; 2 months ago[2] 2.7.15 / 1 May 2018; 40 days ago[3]
Preview release	3.7.0b5[4] / 30 May 2018; 11 days ago
Typing discipline	Duck, dynamic, strong
License	Python Software Foundation License

Figure 6.1: Screenshot Wikipedia: Python



Developer	Apple Inc.
Written in	C, C++, Objective-C, Swift
OS family	Unix-like, based on Darwin (BSD), macOS
Working state	Current
Source model	Closed source
Initial release	June 29, 2007; 10 years ago
Latest release	11.4[1] (15F79)[2] (May 29, 2018; 12 days ago) [±]
Latest preview	11.4.1 Beta 1[3] (15G5054c)[4] (May 30, 2018; 11 days ago) [±] 12.0 Beta 1[5] (16A5288q)[6] (June 4, 2018; 6 days ago) [±]

Figure 6.2: Screenshot Wikipedia: iOS

For this reason, the extraction of the information was designed so that the domain expert specifies the link to the Wikipedia page (e.g. `https://en.wikipedia.org/wiki/Perl`) and further defines, which fields should be extracted by the prototype (e.g. „Latest preview“). The prototype then visits the Wikipedia page on a daily basis, extracts the defined field and compares it against the previously extracted information. If the prototype detects a change, the new value is saved in the database and an update warning is created.

We have selected 68 technologies to evaluate the prototype. From these 68 technologies, 51 have a Wikipedia page which info boxes provided release information. The entire

[1] `https://en.wikipedia.org/wiki/Python_(programming_language)`
[2] `https://en.wikipedia.org/wiki/IOS`

extraction history can be found in the attachment under A.3.

The evaluation of the data source Wikipedia will be accompanied by two examples: the extraction of the field *Stable release* from the Wikipedia page of NodeJS[3] and the extraction of the fields *Stable release* and *Preview release* from the Wikipedia page of Python[4].

For the extraction of the release information of NodeJS, we have configured the URL and the field „*Latest release*“. During the evaluation period from January to June 2018, the prototype was capable of detecting 13 changes, which are listed in Table 6.7.

Release Type	**Extracted Value**
Stable release	9.4.0 & 8.9.4 (LTS) / January 10, 2018
Stable release	9.5.0 & 8.9.4 (LTS) / January 31, 2018
Stable release	9.6.1 & 8.9.4 (LTS) / February 22, 2018
Stable release	9.7.1 & 8.9.4 (LTS) / March 2, 2018
Stable release	9.8.0 & 8.10.0 (LTS) / March 7, 2018
Stable release	9.10.1 & 8.11.1 (LTS) / March 29, 2018
Stable release	9.11.1 & 8.11.1 (LTS) / April 5, 2018
Stable release	10.0.0 & 8.11.1 (LTS) / April 24, 2018
Stable release	10.1.0 & 8.11.1 (LTS) / May 8, 2018;
Stable release	10.1.0 / May 8, 2018;
Stable release	10.2.1 / May 24, 2018
Stable release	10.3.0 / May 29, 2018
Stable release	10.4.0 / June 6, 2018
Stable release	10.4.1 / June 12, 2018

Table 6.7: Wikipedia: Extraction of Field *Stable Release* (January - June 2018)

From this example we can deduce that the monitoring of the release information can detect both minor (e.g. change from 9.4.0 to 9.5.0) and major (e.g. change from 9.11.1 to 10.0.0) releases. The example further demonstrated that the prototype must be capable of handling changes in the way the release information is formulated.

The second example shows the extraction of the release information for Python. In contrast to the first example, we were able to extract the field *Preview release* in addition to the field *Stable release*. The extracted values are listed in Table 6.8.

According to the field *Stable release*, the current stable version of Python is 3.6.5., which coincides with the official website of Python[5]. As mentioned before, the Wikipedia page of Python offers information about the preview release of version 3.7.0 as well. The

[3]`https://en.wikipedia.org/wiki/Node.js`

[4]`https://en.wikipedia.org/wiki/Python_(programming_language)`

[5]`https://www.python.org/downloads/`, visited on 12.06.2018

information about the preview releases are also correct, as the release schedule[6] on the official website of Python shows.

Release Type	**Extracted Value**
Stable release	3.6.4 / 19 December 2017
Preview release	3.7.0a4, 3.5.5rc1, 3.4.8rc1 / 2018
Preview release	3.7.0b1 / 2018
Stable release	3.6.5 / 28 March 2018
Preview release	3.7.0b3 / 29 March 2018
Preview release	3.7.0b4 / 2 May 2018
Preview release	3.7.0b5 / 30 May 2018

Table 6.8: Wikipedia: Extraction of Fields *Latest Release* and *Preview Release* (January - June 2018)

From the second example we can deduce that the field *Preview release* can offer information about an upcoming release, which further helps detecting such technology releases before they get released.

In summary, it can be said that Wikipedia is a good place to extract uniform release information without having to build Web crawlers for various Web sites. The actuality of the release information depends heavily on the Wikipedia community as they are responsible for making adjustments. On the one hand, if a page provides release information, such as latest release or stable release, it serves as a last anchor point in the process of detecting technology releases. On the other hand, if the Wikipedia page offers release information like the *Latest preview* or *Preview release*, this data source may help in the process of detecting technology releases before the actual release date.

6.3 Data Source: Search Engine (Results)

The number of search results is a value, which is displayed after performing a search on Google and it indicates, how many search results were found. We have integrated it into the prototype's Categorization Model for Online Data Sources (see Section 4.2), because of the initial idea that an increasing number of search results indicates a rising interest in the technology. The values are extracted with the help of the Java library Jsoup and is described under Section 5.3.

To examine the established theory, we have collected this value over a period of four months at regular intervals for 94 keywords. The analysis of the resulting time series has shown that the number of search results apparently depends on many factors and is therefore not suitable for trend detection. As an example, we introduce the analysis results for the keyword *Microsoft Windows* in more detail in order to justify the exclusion of this

[6] `https://www.python.org/dev/peps/pep-0537/`, visited at 12.06.2018

data source for the detection of technology trends. Table 6.9 displays selected values to illustrate the unexpected behaviour. It shows that the number of search results increases within 5 days by over 100 times (compare values from 05.03.2018 and 10.03.2018).

Date	**Number of Search Results**
24.02.2018	6.520.000
02.03.2018	11.400.000
04.03.2018	108.000.000
05.03.2018	5.570.000
10.03.2018	640.000.000
12.03.2018	23.600.000

Table 6.9: Analysis of Number of Search Results for Keyword *Microsoft Windows*

Figure 6.3 displays a graphical representation of the number of search results for the same keyword, extracted between February and May 2018. The same findings, namely the arbitrariness of the number of search results, can be drawn.

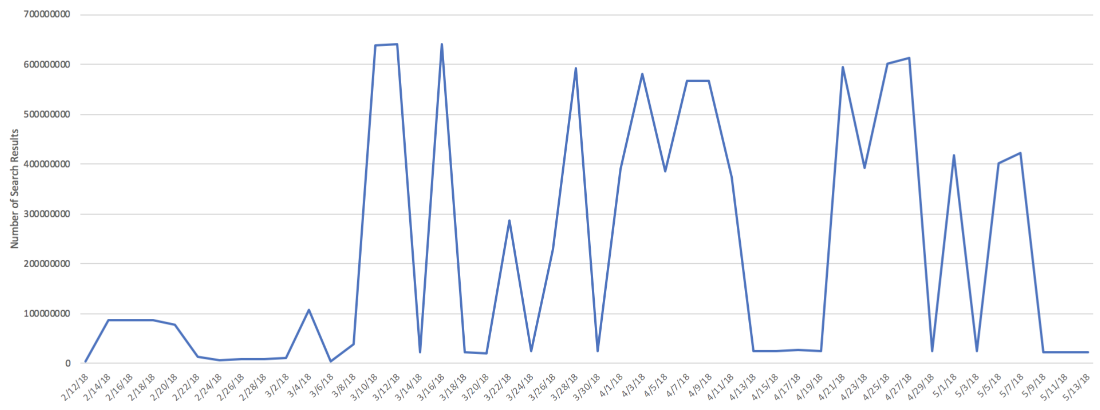


Figure 6.3: Number of Search Results for Keyword Microsoft Windows (112.02.2018 - 13.05.2018)

After further investigations we found several Google support pages (e.g. [53], [54]) which state that the number of search results displayed by Google Search is not an actual, but an estimated amount. Further, the sources mention a request parameter *rc*, *„to request an accurate result count for up to 1M documents, but it might introduce high latency“* [54]. However, we could not achieve more accurate results by using this parameter.

Since the extracted values of the other keywords show similar results, we conclude that the number of search results displayed by Google Search cannot be used to identify trends for technologies due to its unexpected behaviour.

6.4 Data Source: Search Engine (Requests)

In addition to the number of search results extracted from Google Search, the prototype is also capable of extracting the number of search requests, which is gained from Google Trends. Google Trends offers the possibility to analyze the search behaviour, more precisely the number of search requests for specific keywords and by defining some meta data, as described in Section 2.3.2.

Our assumption for the data source was that an imminent update or the announcement of a new release of a technology influence the search behaviour of people, who are interested in this topic. More specific, we assume that the amount of people who are searching for a technology increase and therefore the number of search requests too. This is why the number of search requests was integrated in the prototype's Categorization Model for Online Data Sources (see Section 4.2).

For the evaluation of this part of the prototype, we extracted the Google Trends data over five months for 126 keywords. Each day, the prototype extracted a CSV file per keyword containing the number of search requests for the past 90 days. Google Trends offers normalized data which means that the values range from 0 to 100.

We developed various statistical approaches to analyze the number of search requests. First, we discuss the analysis of the moving average (see Section 6.4.1). Then, the analysis process of technologies containing a weekend trend (see Section 6.4.2) as well as the analysis process of uncommon technologies (see Section 6.4.3) are further investigated. Based on the section about uncommon technologies and their problem of a high amount of null values, we then introduce the analysis of the number of jumps in Section 6.4.5. The last analysis method for the number of search requests is the standard deviation, introduced in Section 6.4.4.

6.4.1 Moving Average

During the analysis of the extracted CSV files containing the number of search requests we found that a significant change of the average over a time period can be a good indicator for news about an (upcoming) technology release. The process of analyzing the averages is described by giving two examples:

The analysis of the exported number of search requests has shown that for wide spread technologies a kind of basic noise exists. For example, the graphical illustration for the search term *iOS 11* in the period from 13.12.2016 to 13.03.2017 looks as follows:

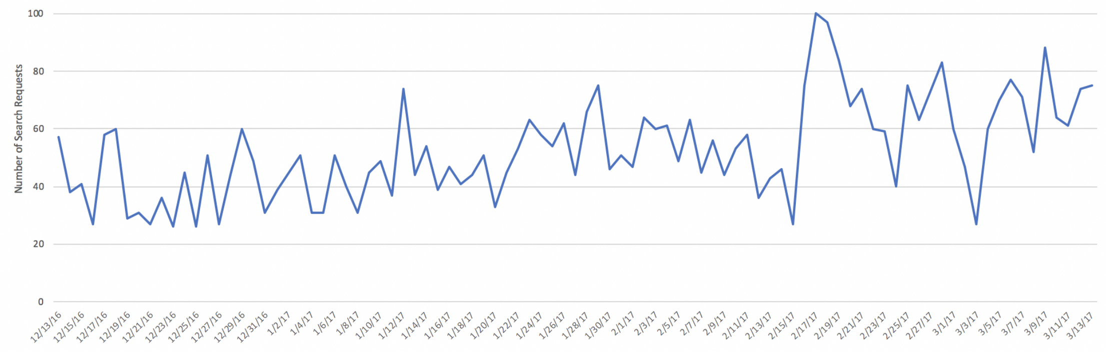


Figure 6.4: Number of Search Requests for Keyword iOS 11 (13.12.2016 - 13.03.2017)

If we repeat the data extraction for the keyword iOS 11 but move the time period two days forward (15.12.2016 - 15.03.2017), a significant increase of search requests is displayed (see Figure 6.5). This may indicate an accumulation of news about an upcoming release of iOS 11. In any case, this sudden increase relativizes the background noise, which causes the graph to change very clearly. The average moves from 52,6 (13.12.2016 - 13.03.2017) to 21,9 (15.12.2016 - 15.03.2017) due the fact that the exported data from Google Trends are normalized and a significant increase of search requests causes a decrease of the number of search requests tracked before. The prototype is capable of detecting such significant changes and creates an update warning which is then forwarded to the domain expert.

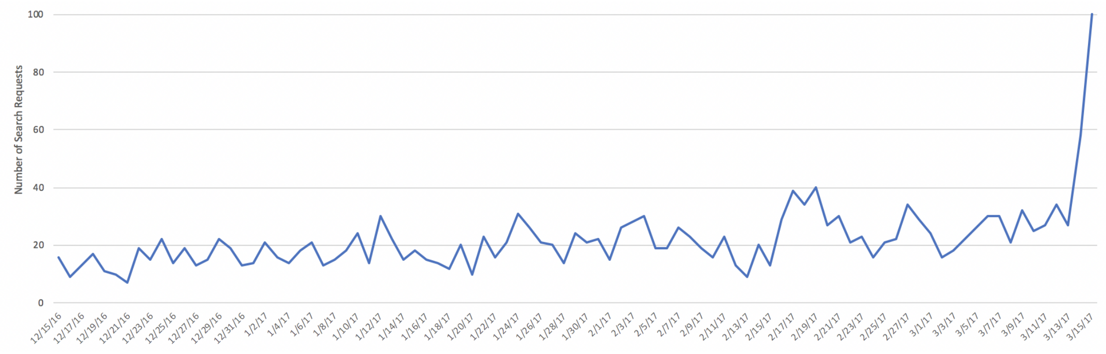


Figure 6.5: Number of Search Requests for Keyword iOS 11 (15.12.2016 - 15.03.2017)

The next two graphs are again the result of the same data extraction process, but here the first graph (see Figure 6.6) represents the number of search requests before the official announcement of iOS 11 (03.03.2017 - 03.06.2017), while the second graph (see Figure 6.7) illustrates the data after the announcement (05.03.2017 - 05.06.2017). Once again, the data collected before (and especially the rise on 13th of March 2017) are relativized by the increase on the 5th of July 2017 and the average has further decreased from 29 to 3,5.

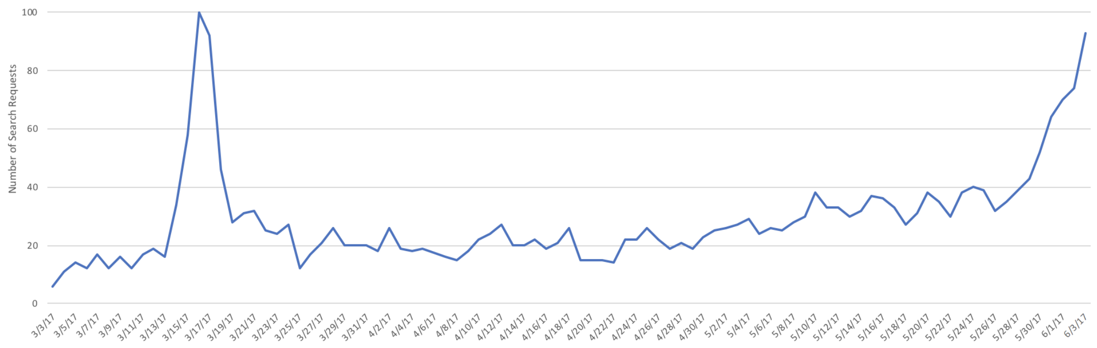


Figure 6.6: Number of Search Requests for Keyword iOS 11 (03.03.2017 - 03.06.2017)

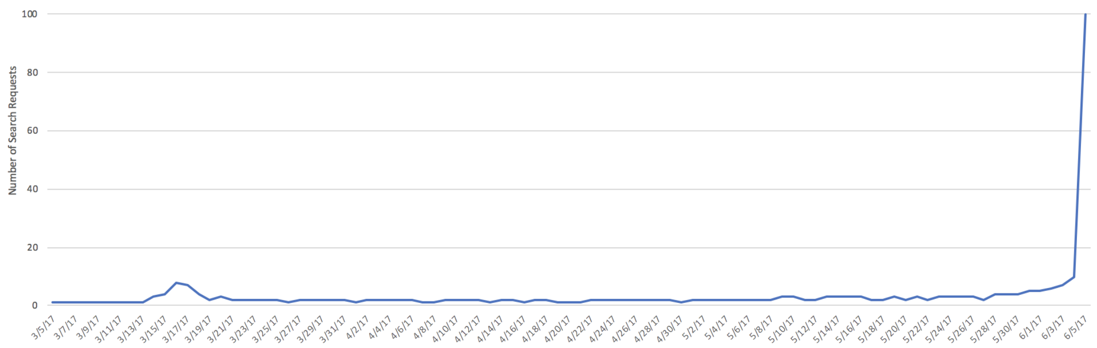


Figure 6.7: Number of Search Requests for Keyword iOS 11 (05.03.2017 - 05.06.2017)

From this example, two things can be derived: on the one hand, the number of search requests increased significantly about four months before the official announcement. From this it can be deduced that the analysis of the search behaviour may detect a technology release at an early stage. On the other hand, the official announcement of a new release can also be detected.

The second example shows the Google Trends data for the keyword *Java 10* and should illustrate the difference between a more widely discussed technology like iOS 11 and a more uncommon one like Java 10. Figure 6.8 shows the number of search requests until two days before the official release of Java 10 (19.09.2017 - 19.03.2018). As before, the graph shows some basic noise including a weekend trend (for more information see Section 6.4.2). A high increase of search requests before the release (as we saw for the keyword iOS 11) was not present.

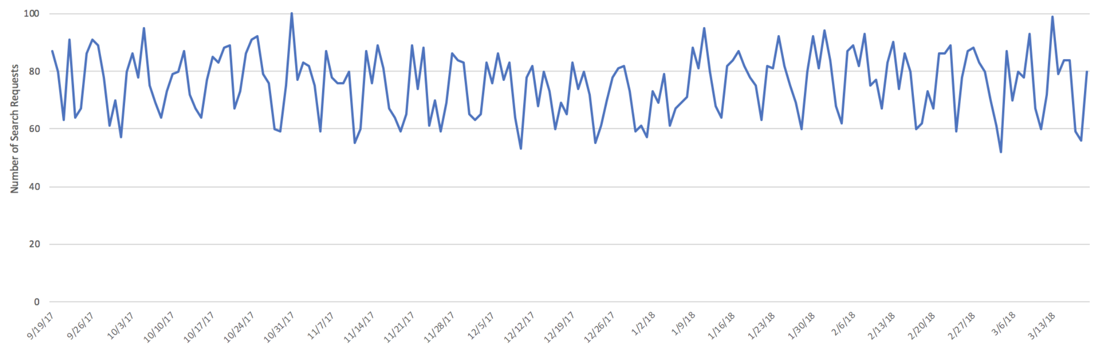


Figure 6.8: Number of Search Requests for Keyword Java 10 (19.09.2017 - 19.03.2018)

Java 10 was released on the 20th of March 2018[7]. The six month extract for the keyword after the announcement looks as follows:

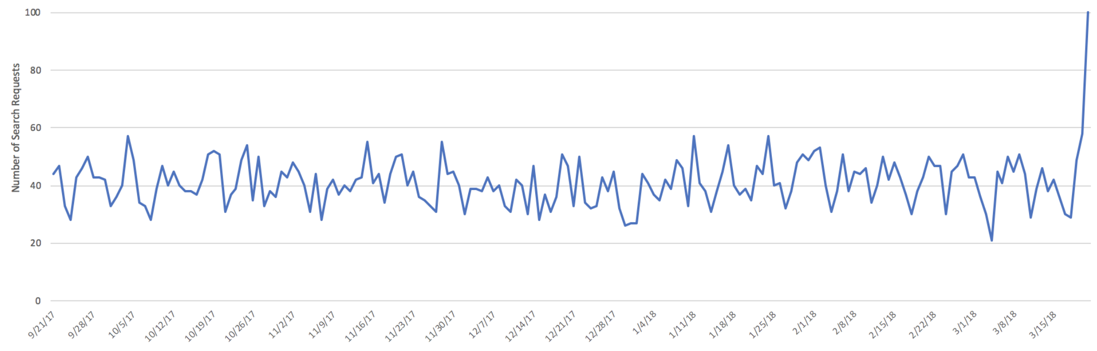


Figure 6.9: Number of Search Requests for Keyword Java 10 (21.09.2017 - 21.03.2018)

Once again, a huge increase of the number of search requests caused a relativization of the averages tracked before. The average decreased from 75,5 to 29. The second example shows that a release increases the interest in the technology and thus reflects in an increasing search behaviour. But, in contrast to the first example, there was no clear increase of search requests ahead of the release, which would have indicated an upcoming release.

In summary, it can be said that the analysis of the normalized data extracted from Google Trends might be helpful to discover a temporary increase of search requests for specific search terms. This may be an indication for an (upcoming) technology release or an accumulation of news about the technology.

Our evaluation has shown that we achieve the most accurate results if an update warning is created, when the current average is 85% or less then the moving average from data

[7]https://www.oracle.com/corporate/pressrelease/Java-10-032018.html

extractions happened before. Further, we do not take keywords into account, which fulfil the restriction with 85% but contain more than 15% zero values, due to the fact that a higher amount of zero values impairs the analysis of the moving average (for more information see Section 6.4.3). In case of many null values, the prototype is performing an analysis of the number of jumps between various data points (see Section 6.4.5). As already suggested by the restriction of zero values, the analysis of the moving average makes more sense for technologies that involve a high search volume.

6.4.2 Technologies with Weekend Trends

The evaluation process of the prototype has focused on technologies which are interesting to ISVs because of the defined research goals (see Section 1.2). These technologies have in common that they are usually more interesting for domain experts than for private individuals. This can lead to the phenomenon that most of the searches are done during the week. A graphical representation of such a technology can be seen in Figure 6.10. It shows the Google Trends data for the keyword Ubuntu from 01.03.2018 to 30.05.2018. The number of search queries is significantly higher during the week than at the weekend, which leads to a graph similar to a sinusoid.

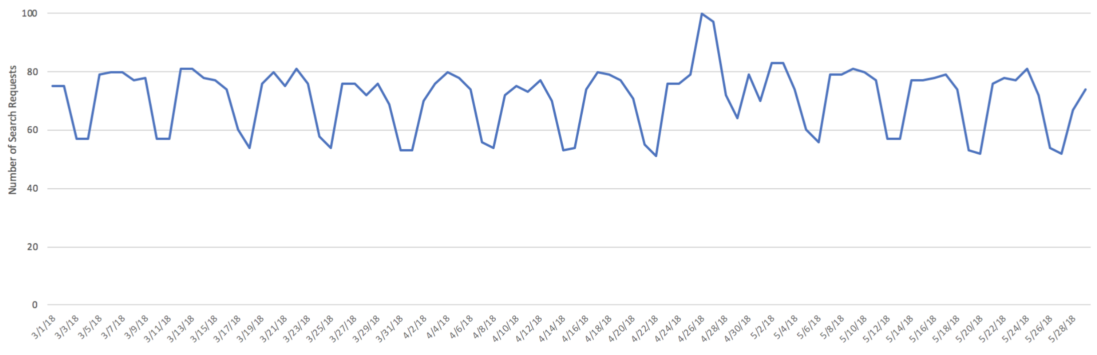


Figure 6.10: Number of Search Requests for Keyword Ubuntu (01.03.2018 - 30.05.2018)

Since such keywords are showing an extreme increase every Monday, there would always be an update warning at the beginning of a week. But the prototype is capable of detecting graphs with weekend trends by calculating the average value of weekdays and weekend days. If there is a significant difference between these two averaging values, a weekend trend is detected, and the graph is handled differently.

Here, the analysis does not focus on the percentage change but on a reduction of a concrete value. Our evaluation process has shown that a decrease by 10 leads to the most accurate results. Again, we skip data extractions which contain more than 15% zero values for the same reason explained in Section 6.4.1.

The analysis of Figure 6.10, however, offers a second insight. At the end of April 2018, the working day values are higher than the values of the weeks before. This is due the fact that Ubuntu has release the newest version of its operation system at 26.04.2018, called Ubuntu 18.04 Bionic Beaver[8], which has caused an increase in search requests.

The release of the new Ubuntu version is even more visible if the expected release version is added to the keyword. The Google Trend data extraction for the keyword *Ubuntu 18* and the same time period as above leads into the following graph:

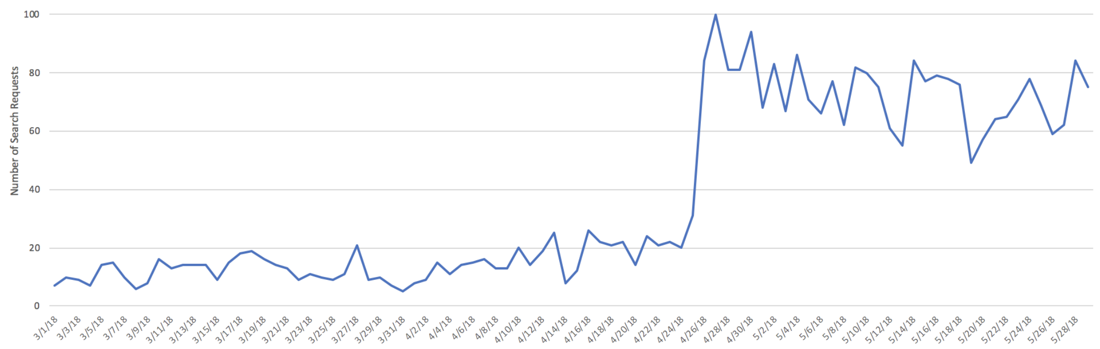


Figure 6.11: Number of Search Requests for Keyword Ubuntu 18 (01.03.2018 - 30.05.2018)

From the analysis of keywords with weekend trend it can be derived that a more precise keyword (Ubuntu vs. Ubuntu 18) may raise the chance of detecting a clearer trend in the data extracted fromm Google Trends.

6.4.3 Uncommon Technologies

In the last two sections, we focused on more common technologies (e.g. Microsoft Windows, Java and PostgreSQL). These technologies are widely used and therefore provide a sufficient number of search requests to analyze the average. However, especially ISVs often use technologies, which are not widely used and the analysis of the averages can be a problem, as we demonstrate be the following example. Figure 6.12 illustrates the Google Trends data for the keyword *BS2000*[9], which is a mainframe operating system developed by Fujitsu.

The chart clearly shows that for the keyword BS2000, Google Trends provides a lot of zero values. These zero values indicate barely existing interest in the keyword. For data extraction like this, none of the previously presented analysis methods can by performed due to the fact that the normalized data points a highly sensible on just a small increase of search requests. This sensibility is also shown by the high number of jumps from high to low values. Thus, these kinds of data extractions usually cannot be used to identify new technology trends. However, the prototype has another method of analyzing time series (see Section 6.4.5), which is capable of handling lots of zero values.

[8]https://www.digitalocean.com/community/tutorials/what-s-new-in-ubuntu-18-04
[9]http://www.fujitsu.com/emeia/products/computing/servers/mainframe/bs2000/

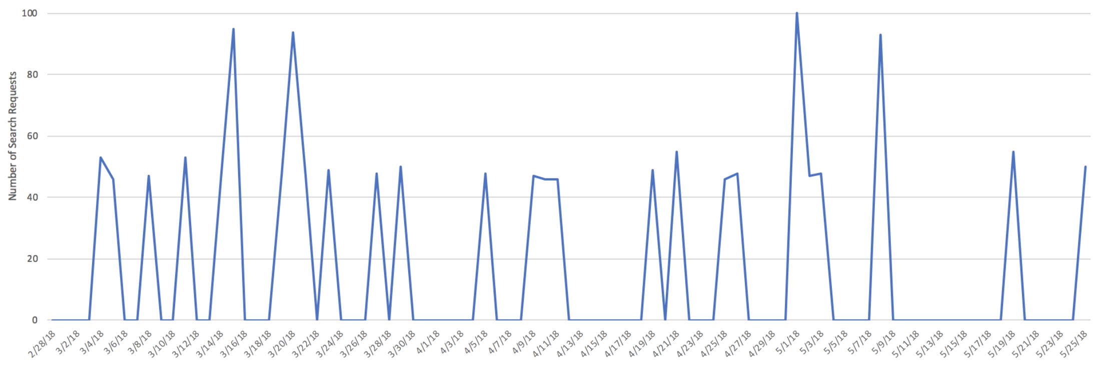


Figure 6.12: Number of Search Requests for Keyword BS2000 (28.02.2018 - 30.05.2018)

6.4.4 Standard Deviation

Besides the analysis of the moving average, the prototype is also capable of analyzing the standard deviation. Similar to the analysis of the average, the standard deviation is calculated for every time series extracted from Google Trends.

The analysis process monitors the changes in the standard deviation across the time series of extraction points. As before, a significant decrease of the standard deviation over time indicates an increase of the search volume, which may be an indicator for an (upcoming) technology release.

The standard deviation analysis described above is now explained by an example: Fedora 28[10], an open-source Linux distribution, was released on the 1st of May 2018[11]. The data extraction from Google Trends for the keyword *Fedora 28* before the release looks as follows:

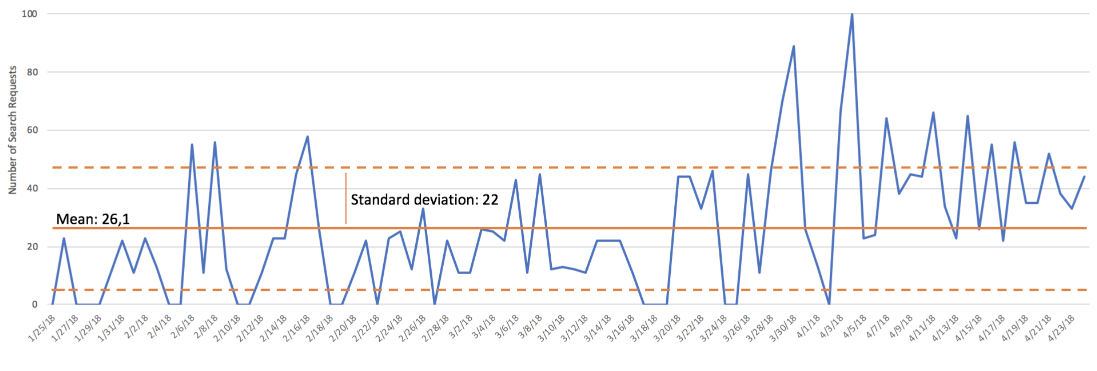


Figure 6.13: Number of Search Requests for Keyword Fedora 28 (25.01.2018 - 25.04.2018)

[10]`https://getfedora.org/`
[11]`https://fedoraproject.org/wiki/Releases/28/Schedule`

The average for the period from 25.01.2018 to 25.04.2018 is 26. The standard deviation is 22. The high value of the standard deviation can be attributed to a lack of a clear trend, which causes high jumps between the days.

Figure 6.14 shows the Google Trends data for the period from 01.02.2018 to 01.05.2018 (release day). The number of search requests changed drastically and so was the graph. The average decreased from 22 to 9,8 and the value of the standard deviation dropped from 22 to 11,9.

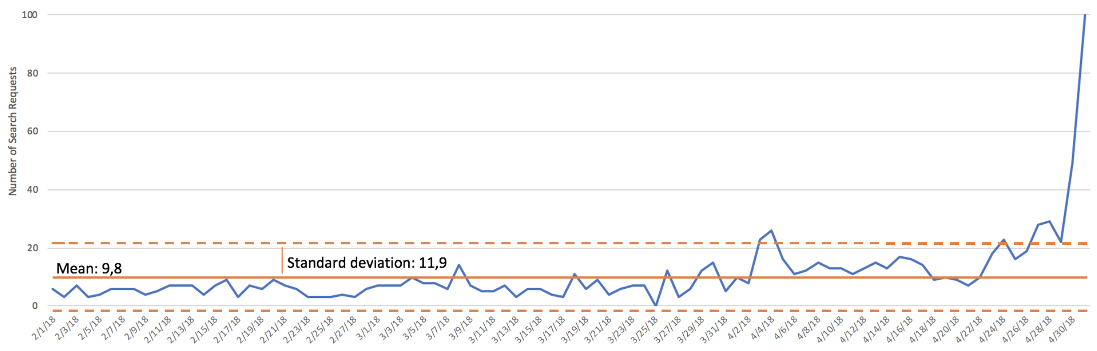


Figure 6.14: Number of Search Requests for Keyword Fedora 28 (01.02.2018 - 01.05.2018)

An in-depth analysis of the extracted values has shown that a decrease of the standard deviation by 20% or more is an accurate threshold for identifying such significant changes. The same threshold has to be fulfilled by the standard deviation of the following two data extractions, to guarantee a clear trend change. Again, uncommon technologies with more than 15% zero values are not considered due to the reasons explained in Section 6.4.3.

6.4.5 Number of high Jumps

Besides the analysis of the moving average and the standard deviation, the prototype is also capable of calculating and analyzing the distance between the various data points from a 90 days extraction. Our expectation was that if the number of jumps decreases significantly, the number of search requests must have increased due to the fact that the extracted data from Google Trends are normalized by 100. The analysis of the number of high jumps was implemented, since the moving average and standard deviation ignore data extractions, which contain more than 15% zero values. This can lead into a problem, especially if the keyword contains a technology and a version number which currently does not exists. Since the version number was not mentioned before, not many people are searching for it which leads into a lot of zero values in the Google Trends extraction. The following example should illustrate the approach of calculating the distance between data points.

Figure 6.15 shows the Google Trends extraction for the keyword *Windows Server 2019* for the period from 20.12.2017 to 20.03.2018. The orange dashed lines illustrate the distance, which is summed up to gain the total number of jumps between the data points. The graph does not reveal any clear trend.

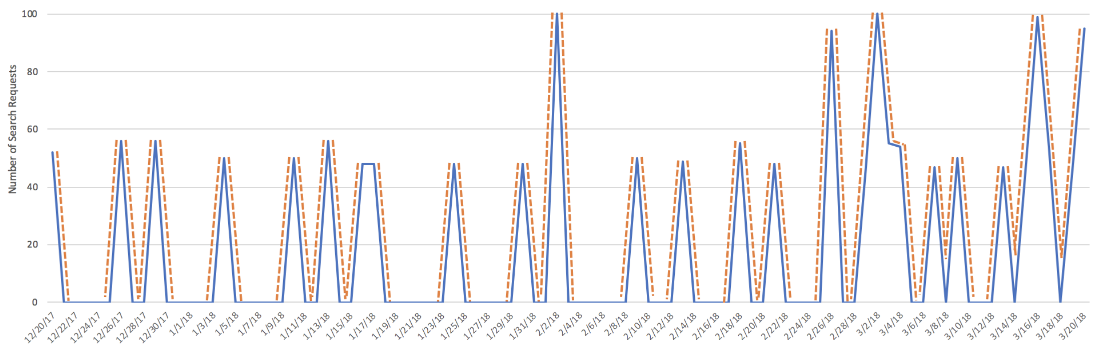


Figure 6.15: Number of Search Requests for Keyword Windows Server 2019 (20.12.2017 - 20.03.2018)

On 20.03.2018, Microsoft released the preview of Windows Server 2019[12]. Figure 6.16 shows the number of search requests for the day after the announcement of the preview. Again, the dramatic increase in search requests relativized the number of search requests that have been tracked before. The average of total jumps from the extractions before was 1258. The current number of jumps was 205, which was a decrease of about 85%.

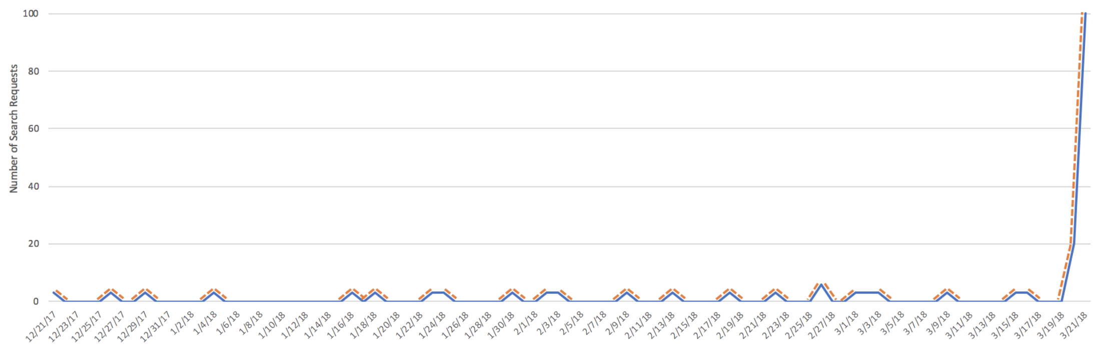


Figure 6.16: Number of Search Requests for Keyword Windows Server 2019 (21.12.2017 - 21.03.2018)

The analysis of the number of jumps has shown that a reduction of 20 percent or more clearly indicates an increased volume of search queries. The analysis has also shown that

[12]https://cloudblogs.microsoft.com/windowsserver/2018/03/20/introducing-windows-server-2019-now-available-in-preview/

this is true, even with 80% of zero values. Thus, analyzing the total number of jumps can be used in addition to the analysis of the moving average and the standard deviation, and is especially useful for unknown keywords, which Google Trends data contains more than 15% zero values.

6.4.6 Compare statistical Approaches

In the last section of the evaluation part, we discuss the various results derived from applying the statistical approaches to the Google Trends data.

Table 6.10 compiles a list of keywords, for which the prototype has detected an increase of search request during the evaluation phase. Overall, through both the moving average and total number of jumps the prototype was capable of detecting nine peaks each. By applying the moving average for keywords which follow a weekend trend, and by calculating the standard deviation, five peaks could be detected by both analysis methods.

Keyword	**MA**	**MAW**	**TNoJ**	**SD**
Android P	✓		✓	
BS2000				
Fedora 28	✓		✓	✓
iOS	✓			
iOS 12	✓		✓	
Java 10	✓	✓	✓	
MacOS	✓	✓	✓	
MySQL 8	✓	✓	✓	✓
NodeJS	✓			
Ubuntu		✓		
Ubuntu 18	✓		✓	✓
WatchOS 5			✓	✓
Windows Server 2019			✓	✓
WordPress 5		✓		
z/OS				

Table 6.10: Google Trends Analysis via various Statistical Approaches; Moving Average (MA), Moving Average Weekend Trend (MAW), Total Number of Jumps (TNoJ), Standard Deviation (DV)

The moving average (MA) has detected two peaks, which have not been detected by other approaches: iOS and NodeJS. From this fact, we deduce that analyzing the moving average is especially helpful, if we do not guess the next version number but still want to monitor the technology.

In addition to the moving average, through the special treatment for data extractions which show a weekend trend (MAW), we were able to detect for both the keyword Ubuntu and WordPress 5 a peak. From this we deduce that handling data extractions with a weekend trend differently may lead into additional acquisition of information.

Analyzing the total number of jumps (TNoJ) is especially helpful, if the system should monitor technologies, for whom the domain expert guessed the next version number. Searching for technologies together with its potential next version numbers can cause a lot of null values, which can be best processed by this analysis method.

Analyzing the data extraction from Google Trends by the standard deviation (SD) has shown that this method succeeds, if the keyword contains the next version number.

In summary, it can be said that by analyzing the Google Trends data with the moving average, the moving average with special treatment for weekend trends, and the total number of jumps, releases can be detected. The found peaks by the standard deviation is a subset of the peaks found by analyzing the total number of jumps. All found increases were attributed to concrete releases. However, each of them was a major and no minor release. Furthermore, for uncommon technologies, such as BS2000 or z/OS, the analysis of the number of search requests is not a suitable method to detect technology updates.

CHAPTER 7

Conclusion

Technologies today have to function in different environments and communicate with different systems, which is why they can no longer be considered as an atomic unit but should be viewed holistically. This aspect can be especially important if an ISV wants to or has to guarantee the compatibility of a system with its environment because of maintenance contracts. It can be a very time- and cost-intensive task to check whether systems of the technology's environment have changed and therefore are no longer compatible. From this circumstance, the main question for this work was derived:

To what extent can the check for technology releases be automated by a system?

In this chapter, we first discuss the main contribution of this thesis by answering the three research questions, comparing the findings against related work, and pointing out the similarities and differences in Section 7.1. Forthcoming research is in turn discussed in Section 7.2.

7.1 Main Contributions

RQ 1: What are promising data sources to detect technology updates?

In order to answer RQ 1, we first identified suitable data sources to build the basis of the data analysis methods. The result was the Categorization Model for Online Data Sources, which defines four different data sources: Technology news, social media, online encyclopaedias and search engines. Technology news, which includes email newsletters and RSS feeds, as well as Twitter, which represent the social media category, are analyzed using NLP. These data sources have been integrated in the data model to find concrete release messages. From Wikipedia (online encyclopedia), we were able to extract concrete release information, such as stable release or latest preview, in a unified way. For search engine category, we extracted two different types of data: (1) the number of search results from Google Search and (2) the number of search requests from Google Trends. The assumption for the search engine data was that if a new technology is released, both the amount of search results and search request increase.

The evaluation of the data analysis methods was performed for each data source independently and has shown that one has to distinguish between minor and major releases when analyzing the data sources. Data sources from the categories technology news, social media and online encyclopedia were capable of detecting both minor and major releases, whereas the number of search requests extracted from Google Trends was just capable of detecting major releases. We developed different analysis methods for the data sources:

Information extracted from the data source categories technology news and social media were analyzed via NLP. The developed NLP pipeline contains two approaches for detecting release messages, which are further compared in RQ 2. Overall it can be said that the analysis of natural language via NLP can aid in detecting technology releases on release date and in advance.

Release information from Wikipedia pages were monitored for changes (see Section 6.2). If the monitored pages offer release information such as "*Stable release*" or "*Latest release*", this data source built the last anchor point in the process of detecting technology releases. However, if a page provides information, such as "*Latest preview*" or "*Preview release*", online encyclopaedias can help detecting upcoming releases as well.

In addition, by analyzing the number of search request extracted from Google Trends with various statistical approaches (see Section 6.4), it is possible to detect an increasing number of people who are searching for a specific technology. As has been shown in Sections 6.4.2, 6.4.5 and 6.4.4, this increase can correlate with actual release announcements. Furthermore, it can also indicate upcoming releases, as was demonstrated by the example in Section 6.4. The comparison of the four statistical analyzing approaches has shown analyzing Google Trends data can help in finding major, but not minor releases (see Section 6.4.6).

In contrast to the findings above, the evaluation of the number of search results extracted from Google Search has shown that this data source cannot be used to detect (upcoming) technology releases. The assumption here was that the amount of web pages reporting of a technology increases, if a new release was announced. However, as the evaluation in Section 6.3 has shown, the number of search results appears to depend on various other factors and can fluctuate substanially, even over two powers of ten within a few days. This unexpected behaviour matches statements from Google, which point out that the number of search results displayed be Google Search is an estimated value and not the exact number of search results.

Summarizing, the data sources selected for our Categorization Model (except the number of search results) can help in both detecting releases at release date as well as in advance. Furthermore, the precision of predictions highly depends on the popularity of the technology, the involved community and whether it is a minor or major release.

RQ 2: To what extent can text processing identify technology updates in texts extracted from heterogeneous online sources?

In order to answer RQ2, we first investigated the extracted data gained from emails, RSS feeds and Twitter to find texts which contain information about (upcoming) releases. The resulting collection of release messages represented the basis for designing and implementing a NLP pipeline (see Section 5.4).

The starting point for NLP tasks is a single sentence, which is why the first task of the NLP pipeline is sentence delimiting. Implementing a proper sentence delimiter was a difficult task and numerous challenges were encountered during development: distinguishing between a dot used in versions number and a dot marking the end of a sentence, and detecting sentences, which do not end with a punctuation character. However, after the successful implementation of the sentence delimiter, we developed two approaches to find release messages through text processing.

The first approach leveraged regular expression to detect release phrases. The second approach aimed at applying certain NLP tools to find concrete release words, such as “publish“, “refresh“, “update“.

In Section 6.1, we evaluated the NLP pipeline in two distinct ways. On the one hand, we compared the two approaches of detecting release messages qualitative and quantitative in Section 6.1.1. On the other hand, we compared the data sources which extract natural language (i.e. emails, RSS feeds and Twitter posts) in terms of simplicity of the extraction process, as well as its timeliness and quality in Section 6.1.2.

The comparison of the two approaches towards detecting release messages has shown both deliver similar results in terms of identified release messages, however, the false positive rate is significantly higher for the NLP toolkit.

The basis of the evaluation of the NLP pipeline consisted of 22.002 Tweets, which contained 746 actual release messages. By the use of regular expressions, the NLP pipeline was capable of detecting 722 of the 746 (96.8%) actual release messages. However, in total 988 Tweets were marked as release messages, resulting in a false positive rate of 26%. The approach of using NLP tools was able to detect 704 of the 746 (94.4%) actual release messages, but achieved a higher false positive rate of 59% (1709 marked Tweets).

Summarizing, by applying NLP to texts extracted from various online data sources, we are capable of detecting release messages. However, an accuracy of 100% cannot be achieved, as words and phrases used in reports about releases can appear in different contexts as well.

RQ 3: What are limitations of automating the release detection process?

In order to answer RQ3, we discuss the degree of automation in the developed Process Model for Technology Update Detection (see Section 4.1) as well as the importance of a domain expert in this system. In the developed system, four main tasks still have to be done by the domain expert: (1) defining technology related keywords, (2) identifying accurate online sources, (3) implementing data extraction methods and (4) giving the system feedback on whether calculated update warnings were valid.

According to the Release Prediction Process Model, the first task for the domain expert is to define technology related keywords. The selection of keywords has a direct impact on the findings of the data analysis methods, regardless on the evaluated data source. For example, if the domain expert correctly suspects the next version number, the number of search requests extracted from Google Trends can deliver clearer results, as seen in the example presented in Section 6.4.2.

In a second step, the domain expert has to identify suitable online sources from which data is to be collected. The number of identified data sources, as well as their quality in terms of accuracy and timeliness have a significant impact on the overall results of the system. For example, selecting relevant email newsletters, RSS feeds and Twitter accounts can lead into better prediction performance of the system. The accuracy of the system can also be increased by defining the correct release information of Wikipedia (e.g. "*Latest preview*", "*Preview release*", etc.).

The third and most time-consuming task the domain expert is required to perform manually is implementing the data extraction methods for the selected data sources.

The last step the domain expert is involved in, is at the end of the release detection process. After the system has found sufficient witnesses of an (upcoming) release, it creates an update warning which is sent to the domain expert. He or she must then decide, whether the update warning is justifiable. The feedback is in turn utilized to adjust the confidence level of the data sources causing the warning.

The other tasks of the Process Model, including information extraction from data sources, analyzing the findings and calculating the release prediction indicator, are automatically performed by the system. This - in contrast to the general monitoring approach where the data sources are checked manually - is the most notable advantage of the developed system.

In summary it can be said that the data mining and analysis step of the Process Model can be performed by a system automatically. However, setting up the system by defining precise keywords, selecting a sufficient amount of data sources which provide qualitative content, and giving the system feedback on its findings are tasks, which we were not able to implement without human interaction. Thus, the developed system provides a semi- but no fully-automated release detection.

Similarities and Differences to Related Work

By answering RQ 1, we have shown that there are several data sources available which can be used to detect technology releases at release date as well as in advance, and that precision highly depends on the popularity of the monitored technology. By answering RQ 2, we have shown that by extracting natural language from various data sources, such as emails, RSS feeds and Twitter posts, we can identify information on (upcoming) releases by processing them via NLP. And, by answering RQ 3, we have shown that our developed Release Prediction Process Model is not capable of detecting technology releases in a fully automated manner. In the following, we proceed to compare the findings of this thesis to related work.

Due to the analysis of the number of search requests extracted from Google Trends, we were able to reveal a correlation between major software releases and an increase in search requests. That the analysis of the search volumes on Google is suitable for the prediction of certain events, coincides with the researched publications introduced in Section 2.3.1. For example, Preis et al. found patterns that may predict stock market movements based on the search behaviour of user of Google Search [50], while Ettredge et al. demonstrated the potential of predicting the unemployment rate in the United States by investigating their behaviour on the Web [17]. However, to the best of our knowledge there is no published work investigating the correlation between software releases and search behaviour of search engine users. By analyzing Google Trends for various technologies, we have shown this data source may help in the process of predicting upcoming major technology releases.

The evaluation of our NLP pipeline has shown analyzing natural language extracted from various online sources via NLP is a suitable tool for finding release messages in texts automatically. To the best of our knowledge, this is the first research which investigated the potential of NLP for detecting release message from natural language. However, since the analysis of natural language is not always clear, the same sentence can lead to different interpretations. Also, the prior knowledge and background of a person can influence the decision making process as to whether it is a release message. Both factors can influence the evaluation of the developed NLP pipeline.

The following publications are linked to the subject of software release prediction in different ways. Their approach, however, focuses more on analyzing data from the development process itself to predict a release date. K. Power introduced in [49] an approach of predicting the likely delivery window of a release in an agile environment by considering the size of the current backlog and the acceptance rate to the date. Papers, such as [24] and [25], concentrating more on software reliability in the prediction process. They proposed the Generalized Software Reliability Model, which successfully predicted the releases of open source software by analyzing the number of faults. Furthermore, Mockus and Weiss conducted an analysis of the risk outgoing from software changes [45]. Here, the focus lies on predicting the failure probabilities caused by changes in the development. However, these publications have in common that their analyses are based on internal factors of the development process, including the amount of issues and the size of the backlog, and so forth, which are normally not accessible to the public. In contrast, our approach is based solely on data which is available to the general public (e.g. email newsletters, social media platforms and Google Trends data).

Finally, applications such as FileHippo App Manager[1], Baidu App Store[2], and Heimdal[3],

[1] https://filehippo.com/

[2] http://pcappstore.baidu.com/

[3] https://heimdalsecurity.com/

which scan for installed programs and check if the latest versions of these programs are installed, are worth mentioning. However, these products only scan for installed software components and not for specified technologies and keywords. Furthermore, they focus more on installing previously released software updates rather than predicting upcoming technology releases.

7.2 Future Research

The result of this thesis is a system, which is capable of extracting data in various forms from various data sources. The evaluation of this system has shown, that email newsletters, RSS feeds and Tweets from Twitter are suitable for finding concrete release messages with the help of NLP. The evaluation of online encyclopaedias has shown that the quality and timeliness of the release information highly depends on the community but can help by detecting technology releases on release date and in advance. The evaluation of the number of search requests has shown that Google Trends data can help in the process of detecting (upcoming) major software releases by analyzing the search volume. However, the future research will focus on the following limitations:

- Expand the Analysis of the Search Engine

  The analysis of the number of search requests is currently focused on a 90-day window, with which we were able to detect major but no minor changes. By changing this time period to 30 or 10 days, it might be possible to detect minor updates as well.

- Connect further Data Sources

  The data source category technology news currently only comprises emails and RSS feeds. In the future, we plan to investigate further data sources, such as blogs, manufacturer websites, as well as other social media platforms, including YouTube and Facebook. Extracting data from question and answer sites, like Stackoverflow[4], or social news sites, like Reddit[5], may provide useful content in terms of release prediction as well. A more technical approach could also be considered. For example, if a system, which should be monitored for changes, offers an API, the latter could be automatically tested for changes.

- Improve NLP Pipeline by Feedback Loop

  The feedback loop of the domain expert, when he or she denied a calculated update warning, can be improved too. The domain expert, for example, could mark the part of the sentence, which leads to a incorrect software warning. This feedback then can be integrated into the NLP pipeline and the false positive rate would decrease by time. This will help to improve the accuracy of the NLP process.

[4] `https://stackoverflow.com/`
[5] `https://www.reddit.com/`

Furthermore, it would be conceivable to leave the field of release detection and test the developed data analysis methods in other environments, since the prediction and trend recognition is also interesting for other areas. For example, current interest in various stocks, cryptocurrencies or funds could be analyzed to develop and adjust appropriate investment strategies.

In summary, it can be said that the developed system is - together with its data sources - capable of detecting release information in natural language and deriving trends from search engine data. However, there are many other fields in which the idea of combining news analyzed by NLP and the search behaviour of search engine users, analyzed by various statistical models, may succeed in detecting trends.

APPENDIX A

Appendix

A.1 Email Newsletter List

Name	**Email Address**	**#**
Adam @ Versioning	versioning@sitepoint.com	120
ADMIN Update	info@admin-magazine.com	34
AIX EXTRA - IBM Systems Magazine	aixextra@e.ibmsystemsmag.com	13
Anand Sanwal	anand.sanwal@cbinsights.com	147
Android Weekly	contact@androidweekly.net	36
Assaf Arkin	assaf@labnotes.org	31
Awesome iOS	newsletter@libhunt.com	373
Azeem Azhar Exponential View	azeem.azhar@exponentialview.co	44
Benedict Evans	list@ben-evans.com	27
blog@cakesolutions.net	blog@cakesolutions.net	57
CA Technologies	info@emea-ca-mail.com	7
Center for Data Innovation	updates@datainnovation.org	39
Cisco	engage@b2me.cisco.com	4
Cisco Press	ciscopress@e.ciscopress.net	3
Claudia Remlinger	claudia.remlinger@neo4j.com	5
Cloud & Data Center Update	DataCenterUpdate@newsletter .windowsitpro.com	29
Computerworld Applications Alert	computerworld_resources@ cwresources.computerworld.com	133
Computerworld Online Resources	online@computerworldmedia.com	79
CRM Buyer	newsdesk@ectnews.com	229
CSS Layout News	me@rachelandrew.co.uk	37

DashingD3js.com	sebastian@dashingd3js.com	6
Data Elixir	lon@dataelixir.com	36
Dave Verwer	dave@iosdevweekly.com	34
DB Weekly	dbweekly@cooperpress.com	32
Designer News	hello@designernews.co	28
Design Systems Weekly	designsystems@designsystems .curatedmail.co	27
Devops Weekly	gareth@morethanseven.net	35
dotNET Weekly	info@dotnetweekly.com	29
Dr Heinz M. Kabutz	heinz@javaspecialists.eu	36
DZone Weekly Digest	mailer@dzone.com	379
Eclipse Foundation	newsletter@eclipse.org	10
Editorial at SDxCentral	newsletter@sdxcentral.com	183
EdSurge	Feedback@edsurge.com	7
EdSurge	feedback@edsurge.com	115
Ember Weekly	info@emberweekly.com	35
ES.next News	hello@esnextnews.com	34
Eugen	eugen@baeldung.com	16
eWebDesign Newsletter Issue #215	newsletter@ewebdesign.com	65
Exasol Xperience 2018	events@exasol.com	5
FeedBurner Email Subscriptions	noreply+feedproxy@google.com	498
Freek Van der Herten	freek@spatie.be	18
Front=2DEnd Front	hello@frontendfront.com	35
Frontend Focus	frontend@cooperpress.com	36
Gamedev.js Weekly	contact@gamedevjsweekly.com	37
Golang Weekly	contact@golangweekly.com	34
Google	no-reply@accounts.google.com	7
Hack Design	contact@hackdesign.org	11
Hack Design	lessons@hackdesign.org	33
Hacker Newsletter	kale@hackernewsletter.com	31
Hacking UI	holla@hackingUI.com	35
Hadoop Weekly	info@dataengweekly.com	19
Hadoop Weekly	info@hadoopweekly.com	11
HP Business	HP@us.mail.hp.com	48
IBM Code	vmdaniel@ibmdeveloperworks .messages3.com	93
IBM i EXTRA - IBM Systems Magazine	ibmiextra@e.ibmsystemsmag.com	11
IBM Systems Magazine	mainframedigital@e .ibmsystemsmag.com	14
IBM Systems Magazine	powersystemsdigital@e .ibmsystemsmag.com	29

IBM Systems Magazine Webinars	webinars@e .ibmsystemsmag.com	103
IDG Connect	IDGConnect@idgconnect-resources.com	62
Infinite Red	newsletters@infinite.red	10
InfoQ	newsletter@mailer.infoq.com	60
InfoWorld Online Resources	online@infoworldmedia.com	21
IT Business Edge Breach Concerns	contentupdates@itbusinessedge.com	258
IT Management Daily	newsletters@itbusinessedge.com	287
ITPro	ITProToday@enews.itprotoday.com	186
ITPro Today	ITProToday@newsletter .winsupersite.com	165
ITPro Update	ITProUpdate@newsletter .windowsitpro.com	35
ITwhitepapers Applications Alert	online_resources@online .itwhitepapers.com	56
ITWhitepapers Applications Alert	online@itwhitepapersmedia.com	11
Jack	jack@jack-clark.net	33
Jakob Nielsen	alertbox@nngroup.com	35
Jakub Chodounský	jakub@csharpdigest.net	36
Jakub Chodounský	jakub@reactdigest.net	35
Jakub Chodounský	jakub@programmingdigest.net	35
Java Magazine	Java@oracleemail.com	4
JavaScript Weekly	jsw@peterc.org	34
JDedwardsERP.com	info@jdedwardserp.com	7
JSter	info@survivejs.com	16
KBall at ZenDev	kball@zendev.com	40
Laravel Daily	info@laraveldaily.com	15
Laravel News	hello@laravel-news.com	38
Linux.com	no-reply@engage.linux.com	4
Linux.com	no-reply@linux.com	28
Linux Foundation Events	no-reply@linuxfoundation.org	5
Linux Journal Newsletter	ljnews@linuxjournal.com	17
Linux Update	info@linux-magazine.com	24
Mainframe EXTRA - IBM Systems Magazine	mainframeextra@e .ibmsystemsmag.com	21
Mainframe Marketplace - IBM Systems Magazine	mainframemarketplace@e .ibmsystemsmag.com	8
Marcelo Ballve	marcelo.ballve@cbinsights.com	35
Mark Needham at Neo4j	devrel@neo4j.com	18
Mark Thomas	markt@apache.org	20
Markus Eisele at Lightbend	m@lightbend.com	3

Mattias Geniar	m@ttias.be	12
Microsoft	Microsoft@e-mail.microsoft.com	17
Microsoft Azure Team	azurenewsreply@microsoft.com	8
MIT Technology Review	newsletters@technologyreview.com	298
MIT Technology Review	promotions@technologyreview.com	48
Mobile Dev Weekly	mobile@cooperpress.com	33
Morning Cybersecurity	morningcybersecurity@politico.com	117
MSSQLTips	newsletter@mssqltips.com	174
MyFavouriteMagazines	future@mail. myfavouritemagazines.co.uk	14
Neo4j Events	emeaevents@neo4j.com	4
Neo4j Webinars	webinar@neo4j.com	6
Node Weekly	node@cooperpress.com	35
NoSQL Weekly	rahul@nosqlweekly.com	38
npm Inc	wombat-jenn@npmjs.com	35
objc.io	mail@objc.io	4
Oracle MySQL	replies@oracle-mail.com	36
O'Reilly Media	oreilly@post.oreilly.com	325
O'Reilly Next:Economy Newsletter	reply@oreilly.com	169
Oren Ellenbogen	oren@softwareleadweekly.com	39
OSNews Newsletter	news@news.nl00.net	252
Outlook Update	OutlookUPDATE@newsletter .windowsitpro.com	29
Phaser World	support@phaser.io	22
Pony Foo	publisher@ponyfoo.com	38
Postgres Weekly	postgres@cooperpress.com	31
Power Systems Extra	powerextra@e.ibmsystemsmag.com	9
Power Systems Marketplace - IBM Systems Magazine	powersystemsmarketplace@e .ibmsystemsmag.com	8
Product Design Weekly by Atomic.io	designweekly@atomic.io	31
Publisher	publisher@linuxjournal.com	7
Pycoders Weekly	admin@pycoders.com	34
Python Weekly	rahul@pythonweekly.com	38
Rachel Nabors	weekly@animationatwork.com	16
Rachel Schallom	rschallom@gmail.com	23
React Newsletter	tyler@tylermcginnis.com	36
Red Hat	email@engage.redhat.com	19
Responsive Design Weekly	justin@responsivedesign.is	38
Rod Trent	ITDevConnections@tech .pentontech.com	8
Ruby Weekly	rw@peterc.org	36

SAP Flash	sap@mailsap.com	43
SAP PRESS	info@news.sap-press.com	83
SDxCentral	research@sdxcentral.com	7
SDxCentral	content@sdxcentral.com	6
SDxCentral	webinars@sdxcentral.com	16
Security Update	SecurityUpdate@newsletter .windowsitpro.com	33
Sharepoint Pro Update	SharePointPro@newsletter .sharepointpromag.com	37
Sidebar	hello@sidebar.io	251
SitePoint Front-end	newsletters@sitepoint.com	21
SitePoint Team	support@sitepoint.com	62
Smashing Magazine	newsletter@smashingmagazine.com	23
Sophia Shoemaker	us@fullstack.io	60
SQLServerCentral.com	subscriptions@sqlservercentral.com	215
SQL Server Pro Update	SQLServerProUPDATE@newsletter .sqlmag.com	34
SUSE	einfo@suse.com	11
SUSE	news@suse.com	5
Swift Weekly	swiftweekly@getrevue.co	28
TechRadar	newsletter@techradar.com	43
The Gradle Team	newsletter@gradle.com	5
The Modern Software Factory	info@na-ca-mail.com	37
The React Newsletter	maurice@theproblemsolver.nl	38
This Week in Rails	newsletter@rubyonrails.org	21
Toolbox General	toolbox@enews.zdb2bmail.com	8
Toolbox Content	content@email.toolbox.com	21
Toolbox Alerts	alerts@email.toolbox.com	81
Toolbox Updates	updates@email.toolbox.com	46
Toolbox Events	events@email.toolbox.com	10
Twitter Info	info@twitter.com	44
Twitter Notify	notify@twitter.com	7
TypeScript Weekly	hello@typescript-weekly.com	33
UI Movement	ramy@uimovement.com	34
Umar Hansa	dev-tips@umaar.com	11
Violeta Georgieva	violetagg@apache.org	8
Vue-newsletter	vuenewsletter@getrevue.co	36
WDRL by Anselm Hannemann	mail@wdrl.info	31
Webdesigner Depot	no-reply@webdesignerdepot.com	22
Web Design Weekly	wdw@jakebresnehan.com	31
WebOps Weekly	peter@webopsweekly.com	34
Web Tools Weekly	submissions@webtoolsweekly.com	34

Windows IT Pro	WindowsITPro@tech.pentontech.com	20
WinInfo Daily Update	WinInfoDaily@newsletter .winsupersite.com	12
wpMail.me	hello@wpmail.me	37
ZDNet Tech Today - US	newsletters@zdnet.online.com	303

Table A.1: List of Email Newsletters

A.2 RSS Feed List

Name	URL	#
Android	https://blog.google/products/android/rss/	25
Apple Release Feed	https://developer.apple.com/news/releases /rss/releases.rss	59
Arch Linux	https://www.archlinux.org/feeds/news/	13
Debian	https://www.debian.org/News/news	7
Fedora	https://fedoramagazine.org/feed/	58
Fujitsu BS2000 OSD	http://ts.fujitsu.com/ps2/press/feed /rss.aspx?id=178	6
HP HP-UX	https://news.hpe.com/rss.xml	61
Linux Mint	https://blog.linuxmint.com/?feed=rss2	16
Microsoft Data Platform Insider	https://blogs.technet.microsoft.com /dataplatforminsider/feed/	25
Microsoft Windows Server	https://blogs.technet.microsoft.com /windowsserver/feed/#	18
Oracle Application	https://blogs.oracle.com/oraclepartners /applications/rss	24
Oracle Database	https://blogs.oracle.com/oraclepartners /database-7/rss	22
Oracle Engineered Systems	https://blogs.oracle.com/oraclepartners /engineered-systems-2/rss	5
Oracle Industries	https://blogs.oracle.com/oraclepartners /industries/rss	9
Oracle Middleware	https://blogs.oracle.com/oraclepartners /middleware/rss	13
Oracle OPN Program	https://blogs.oracle.com/oraclepartners /opn-program/rss	34
Oracle Server and Storage Systems	https://blogs.oracle.com/oraclepartners /server-and-storage-systems/rss	16
Perl	http://feeds.feedburner.com/PerlNews	10
PostgreSQL	https://www.postgresql.org/versions.rss	31
Python	http://planetpython.org/rss20.xml	599

RedHat JBoss	https://www.redhat.com/en/rss/press-releases	36
Ruby	https://www.ruby-lang.org/en/feeds/news.rss	22
Spring Release Feed	https://spring.io/blog/category/releases.atom	44
Ubuntu	https://insights.ubuntu.com/feed	89

Table A.2: List of RSS Feeds

A.3 Wikipedia Data Extractions

Technology	**Release Type**	**Value**
Android	Latest release	8.1.0 "Oreo" / December 5, 2017
Apache Tomcat	Stable release	9.0.8 (May 3, 2018)
Arch Linux	Latest release	Rolling release / installation medium 2018.01.01
Arch Linux	Latest release	Rolling release / installation medium 2018.02.01
Arch Linux	Latest release	Rolling release / installation medium 2018.03.01
Arch Linux	Latest release	Rolling release / installation medium 2018.04.01
Arch Linux	Latest release	Rolling release / installation medium 2018.05.01
Arch Linux	Latest release	Rolling release / installation medium 2018.06.01
C#	Stable release	7.2 / November 15, 2017
C#	Stable release	7.3 / May 7, 2017
C#	Preview release	8.0
C++	Stable release	ISO/IEC 14882:2017 / 1 December 2017
Debian	Latest release	9.3.0 (Stretch) (December 9, 2017)
Debian	Latest release	9.4 (Stretch) (March 10, 2018)
Django	Stable release	2.0.5 (2 May 2018)
Django	Stable release	2.0.6 (1 June 2018)
Ember	Stable release	3.0.0 / February 14, 2018
Ember	Stable release	3.1.0 / April 13, 2018
Fedora	Latest release	27 / 14 November 2017
Fedora	Latest release	28 / 1 May 2018
Fujitsu BS2000 OSD	Latest release	BS2000/OSD v10.0 / 2016
Gradle	Stable release	4.4.1 / December 20, 2017
Gradle	Stable release	4.5 / January 24, 2018
Gradle	Stable release	4.5.1 / February 5, 2018
Gradle	Stable release	4.6 / February 28, 2018

Gradle	Stable release	4.6 (February 28, 2018)
Gradle	Stable release	4.7 (April 18, 2018)
Gradle	Stable release	4.8 (June 4, 2018)
HP HP-UX	Latest release	11i v3 Update 16 / May 2017
IBM AIX	Latest release	7.2 / October 5, 2015
IBM DB2	Stable release	Db2 Data Server (11.1) / April 12, 2016
IBM Informix	Stable release	12.10.xC7 / June 15, 2016
IBM OS400	Latest release	7.3 / April 15, 2016
IBM WebSphere	Stable release	9.0 / June 24, 2016
IBM z/OS	Latest release	Version 2.3 (V2R3) / September 29, 2017
Ingres	Stable release	11.0 / April 21, 2017
iOS	Latest release	11.2.5 (15D60) (January 23, 2018)
iOS	Latest release	11.2.6 (15D100) (February 19, 2018)
iOS	Latest release	11.3.1 (15E302) (April 24, 2018)
iOS	Latest release	11.3 (15E216/15E218) (March 29, 2018)
iOS	Latest release	11.4 (15F79) (May 29, 2018)
Kubernetes	Stable release	1.10.1 / April 12, 2018
Kubernetes	Stable release	1.10.3 / May 21, 2018
Kubernetes	Stable release	1.10.4 / June 6, 2018
Laravel	Stable release	5.6.20 / May 2, 2018
Linux Mint	Latest release	Linux Mint 18.3 "Sylvia" / 27 November 2017
MacOS	Latest release	10.13.3 (17D47/17D2047) (January 23, 2018)
MacOS	Latest release	10.13.3 (17D102/17D2102) (February 19, 2018)
MacOS	Latest release	10.13.4 (17E199) (March 29, 2018)
MacOS	Latest release	10.13.4 (17E202) (April 24, 2018)
MacOS	Latest release	10.13.5 (17F77) (June 1, 2018)
Maven	Stable release	3.5.2 / 24 October 2017
Maven	Stable release	3.5.3 / 8 March 2018
Microsoft SQL Server	Stable release	SQL Server 2017 / October 2, 2017
Microsoft Windows	Latest release	1709 (10.0.16299.248) (February 13, 2018)
Microsoft Windows	Latest release	1709 (10.0.16299.251) (March 5, 2018)
Microsoft Windows	Latest release	1709 (10.0.16299.334) (March 22, 2018)
Microsoft Windows	Latest release	1803 (10.0.17134.1) (April 30, 2018)
Microsoft Windows	Latest release	1803 (10.0.17134.48) (May 8, 2018)
Microsoft Windows	Latest release	1803 (10.0.17134.81) (May 20, 2018)
Microsoft Windows	Latest release	1803 (10.0.17134.81) (May 23, 2018)
Microsoft Windows	Latest release	1803 (10.0.17134.112) (June 12, 2018)
Microsoft Windows	Latest preview	RS4 (10.0.17093) (February 7, 2018)

Microsoft Windows	Latest preview	RS4 (10.0.17101) (February 23, 2018)
Microsoft Windows	Latest preview	RS4 (10.0.17112) (February 23, 2018)
Microsoft Windows	Latest preview	RS4 (10.0.17115) (March 6, 2018)
Microsoft Windows	Latest preview	RS5 (10.0.17627.1000) (March 21, 2018)
Microsoft Windows	Latest preview	RS5 (10.0.17639) (April 4, 2018)
Microsoft Windows	Latest preview	RS5 (10.0.17655) (April 25, 2018)
Microsoft Windows	Latest preview	RS5 (10.0.17661) (May 3, 2018)
Microsoft Windows	Latest preview	RS5 (10.0.17666) (May 9, 2018)
Microsoft Windows	Latest preview	RS5 (10.0.17672) (May 16, 2018)
Microsoft Windows	Latest preview	RS5 (10.0.17682) (May 31, 2018)
Microsoft Windows	Latest preview	RS5 (10.0.17686) (June 6, 2018)
Microsoft Windows Server	Latest release	1709 (10.0.16299) / 17 October 2017
Microsoft Windows Server	Latest preview	RS4 (10.0.17035) / 15 November 2017
MySQL	Stable release	5.7.21 / 15 January 2018
MySQL	Stable release	8.0.11 / 19 April 2018
NEO4j	Stable release	3.3.0 / October 24, 2018
NEO4j	Stable release	3.3.3 / February 12, 2018
NEO4j	Stable release	3.3.5 / April 11, 2018
Net-SNMP	Stable release	5.7.3 / December 8, 2014
NodeJS	Stable release	9.4.0 & 8.9.4 (LTS) / January 10, 2018
NodeJS	Stable release	9.5.0 & 8.9.4 (LTS) / January 31, 2018
NodeJS	Stable release	9.6.1 & 8.9.4 (LTS) / February 22, 2018
NodeJS	Stable release	9.7.1 & 8.9.4 (LTS) / March 2, 2018
NodeJS	Stable release	9.8.0 & 8.10.0 (LTS) / March 7, 2018
NodeJS	Stable release	9.10.1 & 8.11.1 (LTS) / March 29, 2018
NodeJS	Stable release	9.11.1 & 8.11.1 (LTS) / April 5, 2018
NodeJS	Stable release	10.0.0 & 8.11.1 (LTS) / April 24, 2018
NodeJS	Stable release	10.1.0 & 8.11.1 (LTS) / May 8, 2018
NodeJS	Stable release	10.1.0 / May 8, 2018
NodeJS	Stable release	10.2.1 / May 24, 2018
NodeJS	Stable release	10.3.0 / May 29, 2018
NodeJS	Stable release	10.4.0 / June 6, 2018
NodeJS	Stable release	10.4.1 / June 12, 2018
Oracle GlassFish	Stable release	5.0.0 / 21 September 2017
Oracle Java	Stable release	8.0.1610.12 (Update 161) (January 16, 2018)
Oracle Java	Stable release	8.0.1610.12 (Update 161) (January 16, 2018)
Oracle Java	Stable release	8.0.1710.11 (Update 171) (April 17, 2018)
Oracle Java	Stable release	Java 8 (LTS): 8.0.1610.12 (Update 161)

Oracle Linux	Latest release	7.4 / 8 August 2017
Oracle Linux	Latest release	7.5 / 17 April 2018
Oracle MySQL	Stable release	5.7.21 / 15 January 2018
Oracle MySQL	Stable release	8.0.11 / 19 April 2018
Oracle MySQL	Preview release	8.0.4 rc / 23 January 2018
Oracle NoSQL Database	Stable release	18.1 / 20 April 2018 (2018-04-20)
Oracle NoSQL Database	Stable release	4.5 / 24 August 2017 (2017-08-24)
Oracle WebLogic	Stable release	12c R2 (12.2.1) / 26 October 2015
Perl	Stable release	5.26.1 / September 22, 2018
Perl	Stable release	5.26.2 / April 14, 2018
Perl	Preview release	5.27.7 / December 20, 2017
Perl	Preview release	5.27.10 / March 20, 2018
Perl	Preview release	5.27.11 / April 20, 2018
PHP	Stable release	7.2.4 / March 28, 2018
PHP	Stable release	7.2.5 / April 26, 2018
PHP	Stable release	7.2.6 / May 24, 2018
PostgreSQL	Stable release	10.1 / 9 November 2017
PostgreSQL	Stable release	10.2 / 8 February 2018
PostgreSQL	Stable release	10.3 / 1 March 2018
PostgreSQL	Stable release	10.4 / 10 May 2018
Python	Stable release	3.6.4 / 19 December 2017
Python	Stable release	3.6.5 / 28 March 2018
Python	Preview release	3.7.0a4, 3.5.5rc1, 3.4.8rc1 / 2018
Python	Preview release	3.7.0b1 / 2018
Python	Preview release	3.7.0b3 / 29 March 2018
Python	Preview release	3.7.0b4 / 2 May 2018
Python	Preview release	3.7.0b5 / 30 May 2018
RedHat Enterprise Linux	Latest release	7.4, 6.9, 5.11 / August 1, 2017
RedHat Enterprise Linux	Latest release	7.5, 6.9, 5.11 / April 10, 2018
RedHat JBoss	Preview release	7.1 beta / June 29, 2017
RedHat JBoss	Stable release	7.1 / December 13, 2017
Redis	Stable release	4.0.7 / January 24, 2018
Redis	Stable release	4.0.8 / February 2, 2018
Redis	Stable release	4.0.9 / March 26, 2018
Ruby	Stable release	2.5.0 (December 25, 2017)
Ruby	Stable release	2.5.1 (March 28, 2018)
SAP Sybase	Stable release	16.0
Spring	Stable release	5.0.2 / November 27, 2017

Spring	Stable release	5.0.3 / January 23, 2018
Spring	Stable release	5.0.4 / January 23, 2018
Spring	Stable release	5.0.5 / April 3, 2018
Spring	Stable release	5.0.6 / May 8, 2018
Sun Solaris	Latest release	11.3 / October 26, 2015
SuSE Linux Enterprise Server	Latest release	12SP3, 11SP4 / September 7 2017
Swift	Stable release	4.0 / September 19, 2017
Swift	Stable release	4.1 / March 29, 2018
Swift	Stable release	4.1.1 / May 4, 2018
Swift	Stable release	4.1.2 / May 31, 2018
WordPress	Stable release	4.9.5 / 2018-04-03
WordPress	Stable release	4.9.6 / 2018-05-17
WordPress	Stable release	4.9.7 / 2018-05-26

Table A.3: List of Wikipedia Information Extractions

A.4 Twitter Account List

Name	**User name**	**#**
Actian Corporation	@ActianCorp	220
Android	@Android	145
Android Developers	@AndroidDev	225
Angular	@angular	61
Apache Maven	@ASFMavenProject	120
Apache Tomcat	@TheApacheTomcat	55
Canonical Ltd	@Canonical	62
EmberJS	@emberjs	28
Exasol	@ExasolAG	138
Fedora Project	@fedora	181
Fujitsu Global	@Fujitsu_Global	549
GlassFish	@glassfish	30
GNOME	@gnome	158
Google	@Google	1247
Gradle	@gradle	77
HP	@HP	57
HPE	@HPE	296
HPE News	@HPE_News	201
IBM	@IBM	473
IBM Cloud	@IBMcloud	608
IBMDb2	@IBMDb2	397
IBM News Room	@IBMNews	369

IBM Power Systems	@IBMPowerSystems	654
IBM Service Mgmt	@IBMServiceMgmt	436
IBM WebSphere	@IBMWebSphere	265
IBM Z	@IBMZ	747
Java	@java	1203
Kubernetes	@kubernetesio	78
Linux	@Linux	23
Linux.com	@LinuxDotCom	386
Linux Inside: The Ideal Blog for Sysadmins & Geeks	@tecmint	509
Linux Mint	@Linux_Mint	30
Linux Today	@linuxtoday	217
Linux User & Developer	@LinuxUserMag	166
Microsoft SQL Server	@SQLServer	476
MySQL	@MySQL	185
Neo4j	@neo4j	1269
Node.js	@nodejs	504
openSUSE Linux	@openSUSE	322
Oracle	@Oracle	789
php.net	@php_net	23
php.net	@official_php	20
Planet PHP	@planetphp	124
Planet PostgreSQL	@planetpostgres	316
PostgreSQL	@postgresql	205
Python Insider	@PythonInsider	35
Python Software	@ThePSF	299
Red Hat, Inc.	@RedHatNews	1161
Ruby on Rails	@rails	32
SAP	@SAP	664
SAP HANA	@SAPInMemory	163
SAP News	@sapnews	314
Scala	@scala_lang	114
SpringCentral	@springcentral	440
SUSE	@SUSE	950
The Debian Project	@debian	84
The Linux Foundation	@linuxfoundation	1176
The Perl Foundation	@perlfoundation	58
Ubuntu	@ubuntu	409
Windows	@Windows	1332
Windows Server	@windowsserver	127

Table A.4: List of Twitter Accounts data was extracted from

A.5 List of Peaks found by analyzing Search Engine Data

Date	Keyword	Type	#Null	Decrease total	Decrease %
18.02.2018	Android P	MA	0	40 to 32	20%
07.03.2018	Android P	TAoJ	0	920 to 615	33%
02.05.2018	Fedora 28	MA	12	25 to 7	72%
03.04.2018	Fedora 28	TAoJ	9	1972 to 1517	23%
02.05.2018	Fedora 28	SD	12	21 to 13	38%
09.06.2018	iOS	MA	0	74 to 56	24%
09.06.2018	iOS 12	MA	0	50 to 5	91%
09.06.2018	iOS 12	TAoJ	0	959 to 189	80%
25.03.2018	Java 10	MAW	0	71 to 39	45%
24.03.2018	Java 10	MA	0	72 to 48	34%
24.03.2018	Java 10	TAoJ	0	1005 to 776	23%
11.06.2018	MacOS	MAW	0	74 to 42	44%
09.06.2018	MacOS	MA	0	75 to 44	41%
10.06.2018	MacOS	TAoJ	0	675 to 513	24%
11.05.2018	MySQL 8	MAW	2	43 to 25	41%
20.04.2018	MySQL 8	MA	0	47 to 30	36%
24.04.2018	MySQL 8	TAoJ	2	1889 to 1367	28%
20.04.2018	MySQL 8	SD	0	21 to 15	29%
04.05.2018	Nodejs 10	MA	14	29 to 24	17%
30.04.2018	Ubuntu	MAW	0	84 to 72	14%
19.04.2018	Ubuntu 18	MA	0	40 to 34	15%
23.04.2018	Ubuntu 18	TAoJ	0	1747 to 1006	42%
26.04.2018	Ubuntu 18	SD	0	21 to 11	46%
09.06.2018	WatchOS 5	TAoJ	34	1902 to 265	86%
15.06.2018	WatchOS 5	SD	12	23 to 15	33%
25.03.2018	Windows Server 2019	TAoJ	70	1419 to 281	80%
10.06.2018	Windows Server 2019	SD	14	17 to 13	23%
20.02.2018	WordPress 5	MAW	1	47 to 35	26%

Table A.5: Analysis Results of Search Engine Data

List of Figures

Listings

List of Tables

Acronyms

AI	Artificial Intelligence.
API	Application Programming Interface.
CFG	Context Free Grammar.
CLIR	Cross-Lingual Information Retrieval.
CSV	Character-separated Values.
DB	Database.
DOM	Document Object Model.
GIF	Graphics Interchange Format.
HTML	Hypertext Markup Language.
HTTP	Hypertext Transfer Protocol.
IR	Information Retrieval.
ISV	Independent Software Vendor.
NLP	Natural Language Processing.
POP3	Post Office Protocol version 3.
POS	Part-of-speech.
REST	Representational State Transfer.
RSS	Rich Site Summary.
TAM	Technology Acceptance Model.
URL	Uniform Resource Locator.
XML	Extensible Markup Language.

Bibliography

[1] A. M. Aladwani and P. C. Palvia. Developing and validating an instrument for measuring user-perceived web quality. *Information & management*, 39(6):467–476, 2002.

[2] J. F. Allen. Natural language processing. In *Encyclopedia Encyclopedia of Computer Science, 4th Ed.*, pages 1218–1222. John Wiley and Sons Ltd., 2003.

[3] C. Barrière. *Natural Language Understanding in a Semantic Web Context.* Springer, 2016.

[4] B. Berendt, A. Hotho, and G. Stumme. Towards semantic web mining. In *The Semantic Web—ISWC 2002*, pages 264–278. Springer, 2002.

[5] V. Bharanipriya and V. K. Prasad. Web content mining tools: a comparative study. *International Journal of Information Technology and Knowledge Management*, 4(1):211–215, 2011.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[7] M. Cao, Q. Zhang, and J. Seydel. B2c e-commerce web site quality: an empirical examination. *Industrial Management & Data Systems*, 105(5):645–661, 2005.

[8] C. Chapman and K. T. Stolee. Exploring regular expression usage and context in python. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*, pages 282–293. ACM, 2016.

[9] H. Choi and H. Varian. Predicting the present with google trends. *Economic Record*, 88:2–9, 2012.

[10] G. G. Chowdhury. Natural language processing. In *Annual review of information science and technology*, volume 37, pages 51–89. Wiley Online Library, 2003.

[11] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference*, pages 558–567. IEEE, 1997.

[12] R. Cooley, B. Mobasher, and J. Srivastava. Overview of web content mining tools. In *The International Journal of Engineering And Science (IJES)*, volume 2, pages 106–110. CoRR, 2013.

[13] W. B. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*, volume 283. Addison-Wesley Reading, 2010.

[14] M. Cutts. Can you explain what google means by “trust“? `https://www.youtube.com/watch?v=ALzSUeekQ2Q`, 2011. [Online; accessed 12-October-2017].

[15] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw. User acceptance of computer technology: a comparison of two theoretical models. *Management science*, 35(8):982–1003, 1989.

[16] J. Éthier, P. Hadaya, J. Talbot, and J. Cadieux. B2c web site quality and emotions during online shopping episodes: An empirical study. *Information & Management*, 43(5):627–639, 2006.

[17] M. Ettredge, J. Gerdes, and G. Karuga. Using web-based search data to predict macroeconomic statistics. *Communications of the ACM*, 48(11):87–92, 2005.

[18] O. Etzioni. The world-wide web: Quagmire or gold mine? *Communications of the ACM*, 39(11):65–68, 1996.

[19] W. Gatterbauer. Web harvesting. In *Encyclopedia of Database Systems*, pages 3472–3473. Springer, 2009.

[20] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2009.

[21] G. I. Glossary. Isv (independent software vendor). `http://www.gartner.com/it-glossary/isv-independent-software-vendor`, 2017. [Online; accessed 01-August-2017].

[22] Google. How trends data is adjusted. `https://support.google.com/trends/answer/4365533?hl=en&ref_topic=6248052`, 2017. [Online; accessed 17-October-2017].

[23] J. Hogue. pytrends. `https://github.com/GeneralMills/pytrends`, 2017. [Online; accessed 17-October-2017].

[24] K. Honda, H. Washizaki, and Y. Fukazawa. A generalized software reliability model considering uncertainty and dynamics in development. In *International Conference on Product Focused Software Process Improvement*, pages 342–346. Springer, 2013.

[25] K. Honda, H. Washizaki, and Y. Fukazawa. Predicting release time based on generalized software reliability model (gsrm). In *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual*, pages 604–605. IEEE, 2014.

[26] T. Inc. Twitter turns six. https://blog.twitter.com/official/en_us/a/2012/twitter-turns-six.html, 2012. [Online; accessed 12-February-2018].

[27] P. Jackson and I. Moulinier. *Natural Language Processing for Online Applications: Text retrieval, extraction and categorization. Second revised edition.* Natural Language Processing. John Benjamins Publishing Company, 2007.

[28] P. Jackson and F. Schilder. Natural language processing: Overview. In *Encyclopedia of Language & Linguistics (2nd edn.)*, volume 8, pages 503–517. Elsevier Ltd., 2006.

[29] M. Jeong and C. U. Lambert. Adaptation of an information quality framework to measure customers' behavioral intentions to use lodging web sites. *International Journal of Hospitality Management*, 20(2):129–146, 2001.

[30] K. L. Johnson and M. M. Misic. Benchmarking: a tool for web site evaluation and improvement. *Internet Research*, 9(5):383–392, 1999.

[31] R. Kay. Web harvesting. *Computerworld*, 38(25):44, 2004.

[32] J. B. Killoran. How to use search engine optimization techniques to increase website visibility. *IEEE transactions on professional communication*, 56(1):50–66, 2013.

[33] R. Kosala and H. Blockeel. Web mining research: A survey. *ACM SIGKDD Explorations Newsletter*, 2(1):1–15, 2000.

[34] E. Kumar. *Natural Language Processing.* I.K. International Publishing House Pvt. Ltd., 2011.

[35] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massiv Datasets*, chapter 1. Data Mining, pages 1–19. Cambridge University Press, 2011.

[36] C. Library. How do i evaluate websites? https://www.ccri.edu/library/help/evalsites.html, 2016. [Online; accessed 11-October-2017].

[37] C. U. Library. Evaluating web pages: Questions to consider: Categories. http://guides.library.cornell.edu/evaluating_Web_pages, 2017. [Online; accessed 11-October-2017].

[38] E. D. Liddy. Natural language processing. In *Encyclopedia of Library and Information Science (2nd edn.).* Marcel Decker, Inc., 2001.

[39] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data.* Data-Centric Systems and Applications. Springer Berlin Heidelberg, 2011.

[40] C.-J. Luh, S.-A. Yang, and T.-L. D. Huang. Estimating google's search engine ranking function from a search engine optimization perspective. *Online Information Review*, 40(2):239–255, 2016.

[41] S. K. Madria, S. S. Bhowmick, W. K. Ng, and E.-P. Lim. Research issues in web data mining. In *DataWarehousing and Knowledge Discovery*, pages 303–312. Springer, 1999.

[42] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press., 2008.

[43] M. J. Metzger. Making sense of credibility on the web: Models for evaluating online information and recommendations for future research. *Journal of the Association for Information Science and Technology*, 58(13):2078–2091, 2007.

[44] L. Mich, M. Franch, and L. Gaio. Evaluating and designing web site quality. *IEEE MultiMedia*, 10(1):34–43, 2003.

[45] A. Mockus and D. M. Weiss. Predicting risk of software changes. *Bell Labs Technical Journal*, 5(2):169–180, 2000.

[46] NetMarketShare. Desktop search engine market share. `https://www.netmarketshare.com/search-engine-market-share.aspx?qprid=4&qpcustomd=0`, 2017. [Online; accessed 12-October-2017].

[47] T. U. of Edinburgh. How to evaluate website content. `http://www.ed.ac.uk/information-services/library-museum-gallery/finding-resources/library-databases/databases-overview/evaluating-websites`, 2017. [Online; accessed 11-October-2017].

[48] P. M. Polgreen, Y. Chen, D. M. Pennock, F. D. Nelson, and R. A. Weinstein. Using internet searches for influenza surveillance. *Clinical infectious diseases*, 47(11):1443–1448, 2008.

[49] K. Power. Metrics for understanding flow. *Agile Software Development Conference (Agile 2014)*, 2014.

[50] T. Preis, H. S. Moat, and H. E. Stanley. Quantifying trading behavior in financial markets using google trends. *Scientific reports*, 3, 2013.

[51] A. Rosen. Tweeting made easier. `https://blog.twitter.com/official/en_us/topics/product/2017/tweetingmadeeasier.html`, 2017. [Online; accessed 12-February-2018].

[52] J. Srivastava, P. Desikan, and V. Kumar. Web mining – concepts, applications and research directions. In *Studies inFuzziness and Soft Computing*, volume 180, pages 275–307. Springer, 2005.

[53] G. Support. The count of the number of search results is incorrect. `https://support.google.com/gsa/answer/2672285?hl=en`, 2018. [Online; accessed 25-May-2018].

[54] G. Support. Estimated vs. actual number of results. https://support.google.com/gsa/answer/6329272#91428, 2018. [Online; accessed 25-May-2018].

[55] TechTerms. Api definition. https://techterms.com/definition/api, 2016. [Online; accessed 17-October-2017].

[56] M. Tizzoni. j-google-trends-api. https://github.com/elibus/j-google-trends-api, 2017. [Online; accessed 17-October-2017].

[57] P. Trasborg. google-trends-api. https://www.npmjs.com/package/google-trends-api, 2017. [Online; accessed 17-October-2017].

[58] W3Schools. Http methods: Get vs. post. https://www.w3schools.com/tags/ref_httpmethods.asp. [Online; accessed 18-October-2017].

[59] C. N. Wathen and J. Burkell. Believe it or not: Factors influencing credibility on the web. *Journal of the Association for Information Science and Technology*, 53(2):134–144, 2002.

[60] I. Yoon, A. Sussman, A. Memon, and A. Porter. Direct-dependency-based software compatibility testing. In *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, ASE, pages 409–412. ACM, 2007.

[61] I. Yoon, A. Sussman, A. Memon, and A. Porter. Effective and scalable software compatibility testing. In *Proceedings of the 2008 international symposium on Software testing and analysis*, pages 63–74. ACM, 2008.